НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

До захисту допущено

Завідувач кафедри
_____Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
"___" _____ 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системне програмування»

спеціальності **123 «Комп'ютерна інженерія»**

на тему: Засоби управління операціональними трансформаціями для розроблення вебдодатків

Виконав

студент IV курсу, групи КВ-61

Олійник Олександр Валерійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доц. каф. СПіСКС, к. т. н., доцент Тарасенко-Клятченко О.В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ІАЛЦ.045440.001 ОА	Опис альбому	2	
3	А4	ІАЛЦ.045440.002 ТЗ	Технічне завдання	4	
4	А4	ІАЛЦ.045440.003 ТП	Відомість технічного проєкту	2	
5	А4	ІАЛЦ.045440.004 ПЗ	Пояснювальна записка	56	
6	А4	ІАЛЦ.045440.005 Д1	Схема структурна	1	
7	А4	ІАЛЦ.045440.006 Д2	Схема алгоритму	1	
8	А4	ІАЛЦ.045440.007 Д3	Схема структурна	1	
9	А4	ІАЛЦ.045440.008 Д4	Схема структурна	1	

				ДП ІАЛЦ.045440.000		
	ПІБ	Підп.	Дата			
Розробн.	Олійник			Відомість дипломного проєкту	Лист	Листів
Керівн.	Тарасенко-Клятченко				1	1
Консульт.	Клятченко				КПІ ім. Ігоря Сікорського Каф. СПіСКС Гр. КВ-61	
Н/контр.	Клятченко					
Зав.каф.	Романкевич					

Пояснювальна записка
до дипломного проєкту

на тему: Засоби управління операціональними трансформаціями для розроблення вебдодатків

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Системне програмування»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«___»_____2020 р.

ЗАВДАННЯ
на дипломний проєкт студента
Олійника Олександра Валерійовича
(прізвище, ім'я, по батькові)

1. Тема проєкту «Засоби управління операціональними трансформаціями для розроблення вебдодатків»,
керівник проєкту доц. каф. СПіСКС, к. т. н., доцент Тарасенко-Клятченко О.В.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «__» _____2020 р. №1181-С

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту:

- текстовий редактор у вигляді вебдодатку на основі операціональних трансформацій.

4. Зміст пояснювальної записки

- аналіз основних особливостей використання операціональних трансформацій та обґрунтування теми дипломного проєкту;
- аналіз алгоритмів операціональних трансформацій у вебдодатках;
- опис роботи модулів серверної частини вебдодатку;
- тестування серверної частини вебдодатку та аналіз результатів.

5. Перелік графічного матеріалу:

- структура серверної частини вебдодатку (схема структурна);
- взаємодія вебдодатку з віддаленим ОТ сервером (схема алгоритму);
- взаємодія клієнтських сервісів вебдодатку (схема структурна);
- взаємодія модулів серверної частини вебдодатку (схема структурна).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М., к.т.н., доцент каф. СПіСКС		

7. Дата видачі завдання "30" жовтня 2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	18.11.2019	
2.	Розроблення та узгодження технічного завдання	30.11.2019	
3.	Аналіз існуючих рішень	10.01.2020	
4.	Підготовка матеріалів першого розділу проєкту	18.01.2020	
5.	Розроблення програмного забезпечення	16.02.2020	
6.	Відлагодження програмного продукту	14.03.2020	
7.	Підготовка матеріалів другого розділу проєкту	01.04.2020	
8.	Розроблення інтерфейсу програмного забезпечення	20.04.2020	
9.	Підготовка графічної частини	01.05.2020	
10.	Оформлення документації дипломного проєкту	14.05.2020	

Студент _____ Олександр ОЛІЙНИК
(підпис)

Керівник проєкту _____ Оксана ТАРАСЕНКО-КЛЯТЧЕНКО
(підпис)

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (56 с., 25 рис., 3 додатки).

Об’єкт розробки – створення вебдодатку на основі технології операціональних трансформацій, який надає можливість редагувати текстові повідомлення відразу декільком користувачам на основі операціональних трансформацій, передаючи серверу від клієнту всі внесені користувачами зміни для подальшого їх зберігання у вигляді графа.

Розроблений вебдодаток з використанням операціональних трансформацій дозволяє:

- надсилати клієнтські запити на віддалений сервер операціональних трансформацій;
- працювати із простими текстовими документами, навіть декільком користувачам з різних пристроїв одночасно;
- вирішувати клієнтські конфлікти даних;
- застосовувати основні властивості операціональних трансформацій для обробки відразу декількох операцій.

В ході виконання дипломного проєкту:

- розроблено вебдодаток для редагування простих текстових документів, використовуючи ОТ;
- проведено аналіз існуючих рішень;
- розроблено структуру взаємодії клієнта та серверу.

Ключові слова: ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, РОЗПОДІЛЕНІ ОБЧИСЛЮВАЛЬНІ СИСТЕМИ, ОПЕРАЦІОНАЛЬНІ ПЕРЕТВОРЕННЯ, ТЕКСТОВИЙ РЕДАКТОР, ГРАФ КОМІТІВ, JAVA, JAVASCRIPT, ВЕБДОДАТОК, ОТ ТЕХНОЛОГІЯ, ВІДДАЛЕНИЙ СЕРВЕР, КЛІЄНТ.

ABSTRACT

Qualification work includes an explanatory note (56 pages, 25 pictures, 3 appendices).

The object of development is to create a web application based on the technology of operational transformations, which allows to edit text messages to multiple users based on operational transformations, passing to the server from the client all changes made by users for further storage in the form of a graph.

Developed WEB-application using operational transformations allows:

- send client requests to a remote server of operational transformations;
- work with simple text documents, even several users from different devices at the sametime;
- resolve customer data conflicts;
- apply the basic properties of operational transformations to process several operations at once.

During the implementation of the diploma project:

- developed WEB-application for editing simple text documents using OT;
- the analysis of existing decisions is carried out;
- the structure of interaction of the client and the server is developed.

Keywords: SOFTWARE, DISTRIBUTED COMPUTER SYSTEMS, OPERATIONAL TRANSFORMATIONS, GRAPH OF COMMITS, JAVA, JAVASCRIPT, WEB-APPLICATION, REMOTE SERVER, CLIENT.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.002 ТЗ	Засоби управління операціональними трансформаціями для управління вебдодатку. Технічне завдання	4		
	А4	ІАЛЦ.045440.003 ТП	Засоби управління операціональними трансформаціями для управління вебдодатку. Відомість технічного проєкту	2		
	А4	ІАЛЦ.045440.004 ПЗ	Засоби управління операціональними трансформаціями для управління вебдодатку. Пояснювальна записка	56		
	А4	ІАЛЦ.045440.005 Д1	Структура серверної частини вебдодатку. Схема структурна	1		

Змін	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.001 ОА		
Розробив		Олійник О. В.			Засоби управління операціональними трансформаціями для розробки вебдодатків. Опис альбому	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Тарасенко-Клятченко О. В.					НТУУ «КПІ ім. Ігоря Сікорського», ФПМ КВ-61
Н. контроль		Клятченко Я.М.					
Затвердив		Романкевич В.О					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.006 Д2	Взаємодія вебдодатку	1		
			з віддаленим ОТ сервером.			
			Схема алгоритму			
	А4	ІАЛЦ.045440.007 Д3	Взаємодія клієнтських	1		
			сервісів вебдодатку.			
			Схема структурна			
	А4	ІАЛЦ.045440.008 Д4	Взаємодія модулів	1		
			серверної частини			
			вебдодатку.			
			Схема структурна			
		Диск CD-ROM	Текст пояснювальної	1		
			записки.			
			Графічний матеріал			

Змін	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.001 ОА	Арк. 2

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Засоби управління операціональними трансформаціями для розробки вебдодатків».

Галузь застосування: розробка користувацького вебдодатку для редагування простих текстових документів на основі сучасних інформаційних технологій.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проєкту є створення редактору текстових документів, використовуючи особливості операціональних трансформацій.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації в періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

1.1 Вимоги до програмного продукту, що розробляється

- обробка запитів клієнтського програмного забезпечення на основі операціональних трансформацій в рамках клієнт-серверної архітектури;

- застосування основних властивостей операціональних трансформацій;
- сповіщення про виняткові ситуації;
- масштабована архітектура;
- висока відмовостійкість та здатність підтримувати велику кількість клієнтських запитів на сервер;
- наявність графічного інтерфейсу;
- наявність зручної системи управління.

5.2 Вимоги до апаратного забезпечення

- оперативна пам'ять: 4 Гб;
- наявність доступу до глобальної мережі Internet.

5.3 Вимоги до програмного та апаратного забезпечення користувача

- будь-який пристрій зі здатністю відправляти http-запити;

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів
1.	Видача завдання на дипломне проєктування	30.10.2019
2.	Вивчення літератури за тематикою роботи	18.11.2019
3.	Розроблення та узгодження технічного завдання	30.11.2019
4.	Розроблення структури додатку	16.01.2020
5.	Розроблення дизайну та графічних елементів	10.02.2020
6.	Програмна реалізація додатку	16.02.2020
7.	Тестування додатку	14.03.2020
8.	Підготовка матеріалів текстової частини проєкту	01.04.2020
9.	Підготовка матеріалів графічної частини проєкту	01.05.2020
10.	Оформлення технічної документації проєкту	14.05.2020

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.004 ПЗ	Засоби управління операціональними трансформаціями для управління вебдодатку. Пояснювальна записка	55		
	А4	ІАЛЦ.045440.005 Д1	Структура серверної частини вебдодатку. Схема структурна	1		
	А4	ІАЛЦ.045440.006 Д2	Взаємодія вебдодатку з віддаленим ОТ сервером. Схема алгоритму	1		
	А4	ІАЛЦ.045440.007 Д3	Взаємодія клієнтських сервісів вебдодатку. Схема структурна	1		

Змін	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.003 ТП		
Розроб.		Олійник О. В.			Засоби управління операціональними трансформаціями для розробки вебдодатків. Відомість технічного проєкту	Літ.	Аркуш 1 / Аркушів 2
Перевір.		Тарасенко-Клятченко О. В.					
Н. контроль		Клятченко Я.М.				НТУУ «КПІ ім. Ігоря Сікорського», ФПМ КВ-61	
Затв.		Романкевич В.О					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.008 Д4	Взаємодія модулів	1		
			серверної частини			
			вебдодатку.			
			Схема структурна			
		Диск CD-ROM	Текст пояснювальної	1		
			записки.			
			Графічний матеріал			

Змін	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.003 ТП	Арк. 2

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

БД – база даних.

ЕОМ – електронна обчислювальна машина.

ОС – операційна система.

ПЗ – програмне забезпечення.

РОС– розподілена обчислювальна система.

AES (Advanced Encryption Standard) - симетричний алгоритм блочного шифрування.

BLOB (Binary Large Object) – бінарний набір даних, що розглядається як єдиний, неперервний об’єкт.

DNS (Domain Name System) - система перетворення імені хоста (комп'ютера або іншого мережевого пристрою) в IP-адресу.

Git – широко поширене програмне забезпечення контролю версій, яке використовується для роботи з файлами.

Global-OT – технологія роботи з даними, що базується на основі операціональних трансформацій.

IP-адреса (Internet Protocol address) – унікальний ідентифікатор, який використовується для адресації пристроїв у мережі Інтернет з використанням протоколу TCP/IP.

ISP (Internet Service Provider) - Інтернет-провайдер, тобто організація, яка надає послуги доступу та передачі (інформації) через Інтернет.

JSON (JavaScript Object Notation) – текстовий формат обміну даних, оснований на JavaScript, зручний для читання, запису для людини та комп’ютера.

OT (Operational Transformation) – технологія, що використовується для автоматичного вирішення конфліктів у складних системах за допомогою трансформації операцій.

OT-System – система управління даними на основі операціональних трансформацій.

P2P (Peer-to-Peer) – однорангова децентралізована мережа, основана на рівноправності учасників.

React.js – бібліотека мови Javascript для написання вебдодатків.

URL (Uniform Resource Locator) – уніфікований вказівник на ресурс.

WWW (World Wide Web) – всесвітня мережа.

XML (Extensible Markup Language)запропонований консорціумом World Wide Web Consortium (W3C) стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними застосунками, зокрема, через Інтернет.

ВСТУП

На сьогоднішній день доволі широкого застосування набирають системи з можливістю спільного редагування різноманітних документів у режимі реального часу, що дозволяє користувачам вносити свої індивідуальні правки в режимі реального часу і відразу ж бачити модифіковану версію один одного, використовуючи мережу Інтернет.

Однією з головних проблем у створенні систем сумісного редагування є послідовність узгоджень різних виправлень спільних документів під час роботи над певним документом декількома користувачами відразу.

Операціональні трансформації (ОТ) – технологія, яка з самого початку свого існування, була розроблена для підтримки та вирішенню узгоджень при спільному використанні простих текстових документів. Завдяки своїй неблокувальній та ефективній сумісності та необмеженим властивостям взаємодії, ОТ є особливо корисними у мережевих середовищах з високим рівнем затримки та з можливим зникненням підключення до мережі Інтернет. Адже дуже важливим є те, що у разі розриву зв'язку із сервером, у користувача буде відкрита своя локальна копія, яка після відновлення зв'язку з мережею, буде змінена відповідно до глобальної версії текстового файлу, при чому ці зміни будуть узгоджені зі змінами інших користувачів цього файлу.

ОТ все частіше застосовуються в додатках для спільної одночасної роботи, наприклад, Google Wave / Docs, Codoxware, IBM OpenCoWeb і т. д.

Таким чином, вирішено на основі операціональних трансформацій створити вебдодаток, де буде найкраще проілюстровано корисність та важливість використання ОТ.

Вебдодаток матиме можливість одночасного редагування текстових файлів відразу декількома користувачами, що в свою чергу буде слугувати чудовою демонстрацією всіх можливостей ОТ – тобто вирішення конфліктних ситуацій між користувачами під час спільного редагування тексту, а також у разі ймовірного відключення користувача від мережі Інтернет та подальшого

його відновлення з вирішенням ще більшої кількості конфліктних ситуацій. Для зручного користування буде розроблено зручний та інтуїтивно зрозумілий інтерфейс.

1. АНАЛІЗ ОСНОВНИХ ОСОБЛИВОСТЕЙ ВИКОРИСТАННЯ ОПЕРАЦІОНАЛЬНИХ ТРАНСФОРМАЦІЙ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЄКТУ

1.1. Особливості та переваги використання розподілених обчислювальних систем під час користування операціональними трансформаціями

Розподіленою обчислювальною системою називають сукупність комп'ютерів, які є незалежними один від одного та представляються користувачам у вигляді об'єднаної в єдине ціле системи з метою виконання певних обчислювальних задач. В такій системі кожен комп'ютер не залежить від іншого комп'ютера, що свідчить про абсолютну автономність кожного комп'ютера. Отже, можна ствердити, що це надає змогу комп'ютерам незалежно оперувати масивами даних незалежно один від одного, а також зберігати інформацію на них.

Для кожного звичайного користувача розподілена система є нічим іншим як сукупністю однакових комп'ютерів із можливістю збереження, прийому та передачі певних масивів інформації, тобто користувачі не знають яким чином поєднані ці комп'ютери між собою.

Варто відмітити, що до основного компоненту розподіленої обчислювальної системи слід віднести цілий комплекс програмного забезпечення, який слугує для коректної та високопродуктивної роботи програмних додатків, які працюють під керуванням різноманітних операційних систем (рис. 1.1).

Основу такого програмного забезпечення складає цілий ряд обслуговуючих, або ж сервісних, компонентів цього програмного забезпечення, яке продукує взаємодіючий інтерфейс різних програмних

застосунків та розв'язує ряд проблем, які виникають через відмінність програмних та апаратних рішень різних платформ.

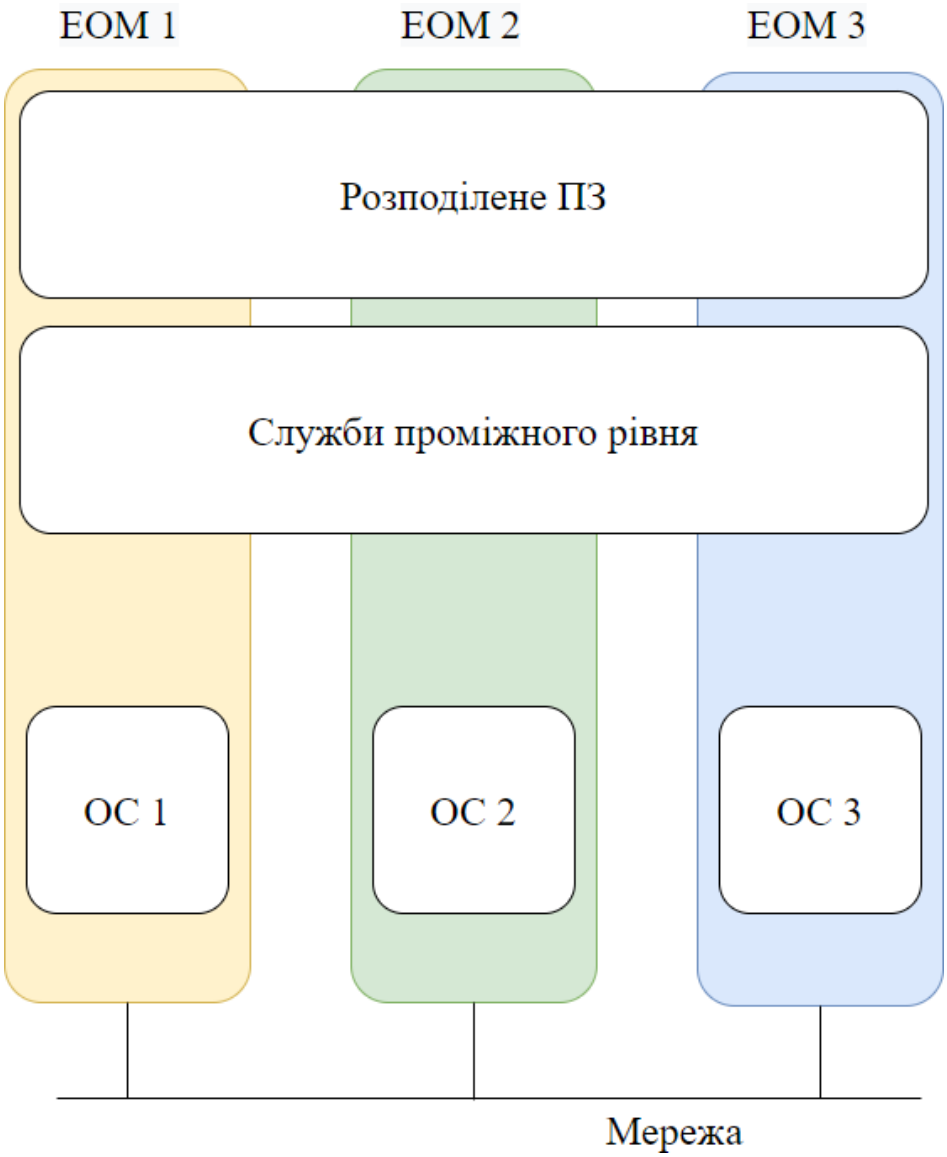

Рисунок 1.1 – Розподілена обчислювальна система із використанням служб проміжного рівня

Головними особливостями розподілених обчислювальних систем є [1]:

- відкритість розподіленої обчислювальної системи до покращень та застосувань, тобто система, є простою для використання користувачами;
- багатоплатформність або ж кросплатформність – здатність програмного забезпечення працювати на різних апаратних та

програмних платформах, що значно зменшує ціну на виробництво нових версій;

- паралельність виконання обчислень, яка здобувається завдяки паралельному надсиланню запитів виконання різноманітних задач на певних, зв'язаних між собою обчислювальних машинах;
- прозорість - це характеристика, при якій від користувача повністю прихована структура та реалізація роботи розподіленої обчислювальної системи. Тобто приховано всі деталі роботи апаратного та програмного забезпечення, а також прихована наявність паралельності виконання обчислень від користувача. Також від користувача повністю приховано всю інформацію щодо доступу до ресурсів системи, виникнення помилок, завдяки чому у користувача даної системи складається враження безперебійної роботи і т.д.;
- використання такого прийому як міграція, тобто перенесення певних ресурсів системи до іншої обчислювальної машини цієї ж самої системи прямо під час виконання певних обчислювальних задач без видимості виконання цих процесів користувачеві;
- масштабованість – це характеристика, завдяки якій при недостатній обчислювальній потужності системи, можна підвищити обчислювальну потужність шляхом нарощення нових програмних та апаратних ресурсів;
- висока відмовостійкість – у разі відмови роботи одного компоненту системи, інші робочі вузли обчислювальної системи можуть швидко почати продовжувати виконувати попередню задачу, без явних проблем в стабільності роботи потрібних задач користувача, тим самим розподілити навантаження системи на різні робочі вузли.

Варто також перелічити цілий список головних переваг розподілених обчислювальних систем, якими керуються при виборі саме систем такого типу [2]:

● високий рівень оптимізації користування наданими ресурсами системи, завдяки підключенню в робочий процес тих компонентів розподіленої обчислювальної системи, які безпосередньо не долучено до активної роботи – центральні та графічні процесори, оперативні та постійні запам'ятовуючі пристрої, периферійні пристрої і т. д.;

● збільшення швидкодії операції доступу до даних за рахунок застосування розподілених БД та широкомовних запитів та відповідей в таких системах;

● високий рівень надійності РОС - при неможливості виконання певної задачі одним модулем, РОС понизить загальний рівень своєї продуктивності, але не свою функціональність, адже інші модулі системи візьмуть на себе роботу модуля, який вийшов з ладу;

● високий рівень швидкості обробки інформації, що в свою чергу підвищує загальну продуктивність обчислювальної системи. Цей ефект досягається за рахунок децентралізованої обробки даних.

Головним та найвідомішим прикладом РОС могло б бути World Wide Web або ж «всесвітнє павутиння». Ця всесвітня мережа надає просту та цілісну модель розподілених документів [3].

Для відкриття потрібного документа користувачеві досить перейти за необхідним посиланням й переглянути необхідну інформацію. Після цього документ з'являється на екрані. Зараз немає необхідності знати, з якого сервера надходить документ, досить лише інформації про те, де він розташований. Сам процес публікації документа є досить тривіальним: необхідно лише задати йому унікальне ім'я в формі уніфікованого покажчика ресурсу - URL, яке посилається на локальний файл, що знаходиться на постійному запам'ятовуючому пристрої з вмістом документу. За умови, якщо б WWW користувачам була б представлена у вигляді величезної централізованої системи документообігу, її можна було б вважати також РОС.

Тому користувачі усвідомлюють, що документи знаходяться в різних місцях і розподілені по не на одних і тих же самих серверах.

1.2. Перелік та аналіз основних завдань РОС

Основним та головним завданням розподілених обчислювальних систем є надання та забезпечення користувачам стабільного доступу до віддалених ресурсів, а також забезпечення їх спільного використання, виконуючи правильне регулювання цим процесом. В загальному випадку ресурси бувають віртуальними, проте часто вони можуть включати в себе комп'ютери, обчислювальні та графічні потужності, необхідні для користувачів, пристрої зберігання даних та багато інших пристроїв, які можна пристосувати до спільного користування. До цього списку також можна віднести вебсторінки, адже вони фактично є файлами, доступ до яких відбувається за активним посиланням.

Існує велика кількість причин для використання спільних ресурсів. Високий рівень економічності є найважливішою та найпоширенішою з них. Прикладом, який найкраще демонструє цю причину, є доступ користувачів до одного спільного принтера, адже таке використання є набагато вигіднішим, аніж покупка окремого принтеру для кожного користувача, який його потребує. З аналогічної причини широкого застосування набуло спільне користування такими розподіленими системами, як суперкомп'ютери або високопродуктивні сховища даних [3].

Взагалі з’єднання ресурсів та користувачів дуже полегшує якість та швидкість обміну інформацією, що в свою чергу підвищує рівень кооперації користувачів. Найбільш яскравим прикладом цього є успіх мережі Інтернет, адже завдяки використанню різноманітних протоколів для обміну даних, фактично весь світ в тому чи іншому вигляді пройшов етап діджиталізації різнопланових робочих процесів – починаючи від звітності в урядових

установах і закінчуючи використанням хмарних обчислень на суперкомп'ютерах, використовуючи лише підключений до мережі Інтернет комп'ютер.

В сучасному світі можливість користування Інтернетом привела до появи великої кількості організацій, які тісно поєднані між собою, але знаходяться на різних континентах планети, й відповідно їх працівники працюють разом над спільною роботою, використовуючи різні системи групової обробки даних та передачі даних: сервіси та програми для редагування документів, проведення онлайн-конференції і т. д. Так само варто згадати про величезну кількість комерційних онлайн-сайтів, на яких є можливість замовлення різноманітних товарів та послуг, не виходячи з дому, що в теперішніх реаліях світу є дуже важливим аспектом життя сучасної людини.

Проте з поступовим зростанням числа підключень, а також рівня кількості використання спільних ресурсів, досить важливими стають питання безпеки. Зараз системи такого виду мають невисокий рівень захисту від вторгнень до ліній зв'язку. Конфіденціальна інформація в багатьох випадках може пересилатися по мережах без використання різних типів шифрування, тобто відкритим текстом.

Також серйозною проблемою в питані безпеки є простеження за користувачами, завдяки якому користувачам пропонується конкретна контекстна реклама, що є порушенням прав особистості, адже це проводиться без прямого сповіщення користувача. Також зараз дуже поширена розсилка так званих спам-листів, від якої можна захиститися лише використовуючи певні спеціальні інформаційні фільтри, які будуть сортувати повідомлення такого типу.

Отже, можна ствердити, що на сьогоднішній день неможливо побудувати та створити щось ідеальне, адже час йде з набагато більшою швидкістю, аніж розвиток технологічного надбання людства.

1.3. Постановка задачі дипломного проєкту

На сьогоднішній день серед різноманітних пропозицій вебдодатків існує величезна конкуренція та ще й їхня кількість збільшується з кожним днем, а користувач повинен обирати серед них найкращий продукт.

В першу чергу користувачеві від вебдодатку потрібні лише високий рівень надійності та швидкодії, а також так званий “User-friendly” інтерфейс, тобто максимально зручний та зрозумілий інтерфейс. Найважливішою та найскладнішою частиною кожного додатку є реалізація саме серверної частини додатку для зв’язку в мережі, адже саме тут відбувається та сама реалізація вищезгаданих надійності та швидкодії.

Задачею бакалаврського проєкту є створення вебдодатку для спільного онлайн редагування текстових файлів з використання алгоритмів ОТ, які будуть вирішувати конфліктні ситуації, що можуть виникати при користуванні цим додатком, а також аналіз роботи операціональних трансформацій.

1.4. Висновки

У даному розділі проаналізовано всі особливості та переваги розподілених обчислювальних систем, обґрунтовано використання РОС під час виконання обчислювальних операцій. Також проаналізовано та перелічено основні задачі розподілених обчислювальних систем.

2. АНАЛІЗ АЛГОРИТМІВ ОПЕРАЦІОНАЛЬНИХ ТРАНСФОРМАЦІЙ У ВЕБДОДАТКАХ

2.1. Особливості сумісного редагування документів та використання ОТ на практиці

Можна ствердити, що сумісна робота є досить складною з технічної точки зору, тому що декілька людей можуть вносити різні зміни в один і той же фрагмент тексту майже в однакові моменти часу. Навіть в ідеальних умовах та з високою швидкістю передачі даних, передача на сервер внесених змін не може здійснюватися миттєво (варто зазначити, що швидкість передачі даних має певні обмеження з технічної точки зору), тому для імітації миттєвого відгуку в кожен момент часу кожен клієнт працює з локальною версією (або реплікою) редагованого документа, яка може відрізнятися від версій інших учасників. І основним завданням сумісної роботи над документом є забезпечення коректної узгодженості між локальними версіями, або ж іншими словами – необхідно забезпечити, щоб всі локальні версії рано чи пізно сходилися в одну і ту ж коректну версію документу (вимагати того, щоб всі клієнти в певні моменти часу одночасно мали одну і ту ж версію є досить некоректно, так як процес редагування може бути нескінченним) [4].

Найпростішим варіантом рішення такої проблеми є просте порівняння вмісту різних версій документу. Розглянемо це на наступному прикладі.

Нехай є два клієнта – Клієнт 1 та Клієнт 2, й поточний стан документу є синхронізованим та відомим обом. Сервер при отриманні повідомлення про зміну документу від Клієнта 1 знаходить різницю між його версією та своєю й намагається поєднати дві версії в одну найефективнішим доступним способом. Потім сервер надсилає оновлену версію Клієнту 2. Якщо Клієнт 2 має якісь ненадіслані до серверу зміни, то процес буде повторюватися, адже

необхідно поєднати версію від сервера з локальною копією Клієнта 2 та відправити знову на сервер.

Дуже часто таке рішення працює не зовсім коректно. Наприклад, нехай Клієнт 1, Клієнт 2 та сервер починають свою роботу над синхронізованим документом, який містить текст “Рудий пес” (рис. 2.1).

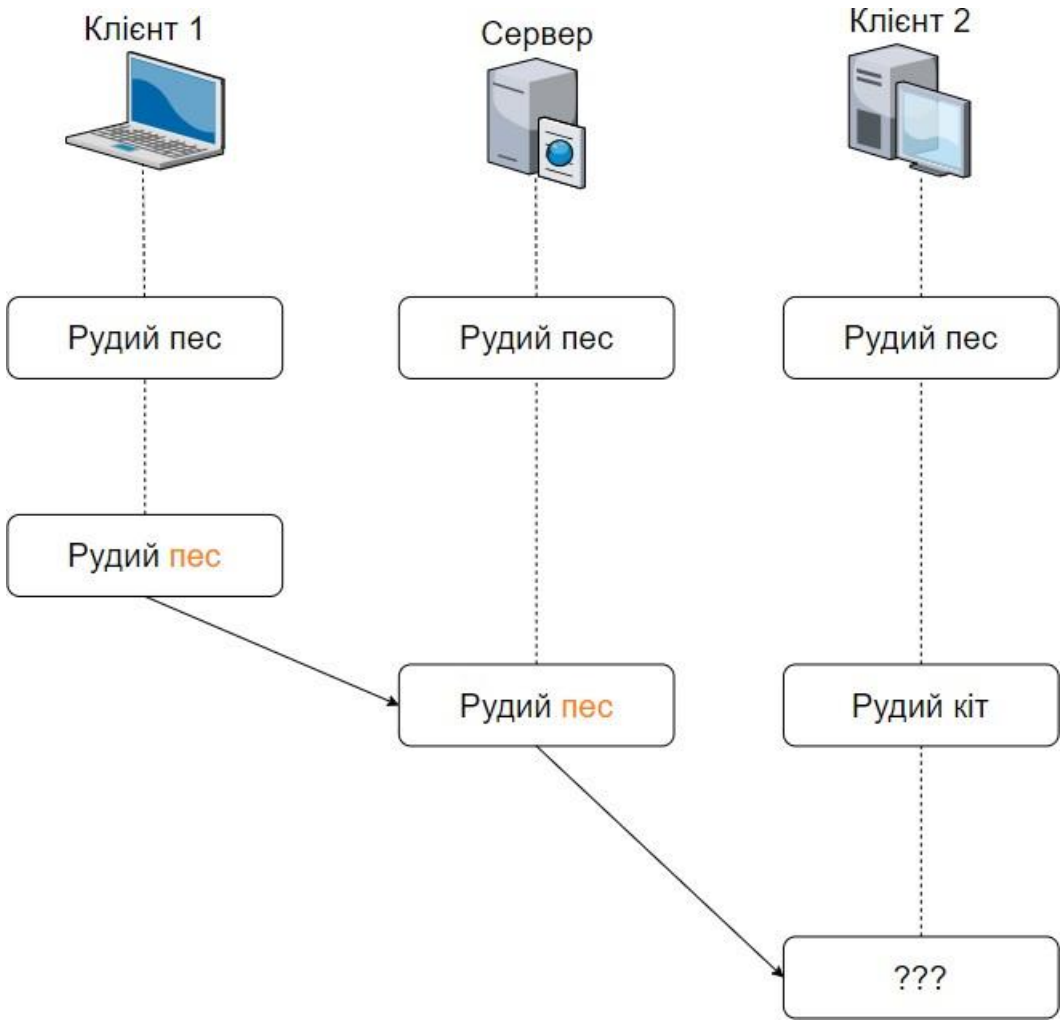

Рисунок 2.1 – Приклад роботи додатку, який реалізовує сумісну роботу над текстовим документом на основі порівняння вмісту різних версій документу

Клієнт 1 змінює колір слова “пес”, в той час як Клієнт 2 змінює слово “пес” на “кіт”. Припустимо, що зміни Клієнта 1 було надіслано першими до

серверу. Тоді сервер повинен надіслати змінену версію до Клієнта 2, і тут вже не зовсім зрозуміло яка версія повинна відображатися у Клієнта 2, адже варіанти “Рудий пес”, “Рудий кіт” та “Рудий пес кіт” (у варіаціях зі зміненими кольорами та сталим чорним кольором), в принципі є коректними, але не зовсім вірними, найправильнішим повинен бути рядок “Рудий кіт” зі виконаною зміною кольору.

В таких випадках така поширена система, як Git, повідомить користувача про помилку й необхідно буде виконувати зміни вищезгаданого типу в ручному режимі. В цьому і полягає головний недолік такого підходу, адже користувач ніколи не може бути абсолютно впевненим в правильності внесених змін до документу.

В теорії можна, наприклад, вирішити це блокуванням тієї частини тексту, яку вже хтось редагує, для інших користувачів, але це не є правильним рішенням, так як це може доволі сильно погіршити досвід роботи користувача з даним застосунком. Тим паче, можливий варіант, коли відразу два користувача почали редагувати однакову частину тексту й один із них може або втратити свої внесені до документу зміни, або ж йому прийдеться вручну ще раз їх вносити.

Тепер розглянемо роботу додатку, в якому документи зберігаються у вигляді послідовності змін, тому замість порівняння вмісту документу потрібно послідовно застосовувати необхідні зміни. Знаючи зміни, які виконував користувач до текстового документа, є можливість коректно визначити правильну кінцеву версію файлу після застосування всіх необхідних змін.

В загальному випадку є три різні операції:

- вставка тексту;
- видалення тексту;
- застосування різних стилів до тексту (переведення тексту в курсив, зміна кольору і т. д.).

Отже, коли відбувається редагування документа, всі виконані дії над документом зберігаються в конкретній послідовності операцій, вони заносяться в кінець журналу змін. При відображенні документу всі дії з цього журналу будуть виконані [5].

Тепер розглянемо наступний приклад.

Нехай Клієнт 1 та Клієнт 2 починають свою роботу над синхронізованим документом, який містить текст “КПІ 1898”. Клієнт 1 здійснює наступну операцію над текстом – змінює “КПІ 1898” на “КПІ 1899”, і насправді це дві операції – видалення та вставка нового елементу (рис. 2.2).

	Документ Клієнта 1							
Індекси	0	1	2	3	4	5	6	7
	К	П	І		1	8	9	8
Видалення на поз. 7	К	П	І		1	8	9	
Вставка '9' на поз. 7	К	П	І		1	8	9	9

Рисунок 2.2 – Операції Клієнта 1

Тим часом, Клієнт 2 виконує наступні зміни в документі – змінює текст на “НТУУ КПІ 1899” (рис. 2.3).

	Документ Клієнта 2												
Індекси	0	1	2	3	4	5	6	7	8	9	10	11	12
	К	П	І		1	8	9	8					
Вставка 'Н' на поз. 0	Н	К	П	І		1	8	9	8				
Вставка 'Т' на поз. 1	Н	Т	К	П	І		1	8	9	8			
Вставка 'У' на поз. 2	Н	Т	У	К	П	І		1	8	9	8		
Вставка 'У' на поз. 3	Н	Т	У	У	К	П	І		1	8	9	8	
Вставка ' ' на поз. 4	Н	Т	У	У		К	П	І		1	8	9	8

Рисунок 2.3 – Операції Клієнта 2

Клієнту 2 приходить з журналу змін операція видалення, яку виконав Клієнт 1, якщо ж Клієнт 2 її виконає, то вийде помилковий варіант тексту (рис. 2.4).

	Документ Клієнта 2												
Індекси	0	1	2	3	4	5	6	7	8	9	10	11	12
	Н	Т	У	У		К	П	І		1	8	9	8
Видалення на поз. 7	Н	Т	У	У		К	П		1	8	9	8	

Рисунок 2.3 – Виконання некоректної операції Клієнтом 2

Для запобігання таких помилкових ситуацій Клієнт 2 повинен виконати трансформацію операцій (1) та (2), які виконав Клієнт 1 у себе, у відповідності до поточних змін, які було внесено Клієнтом 2, та виконати вже нову перетворену операцію (рис. 2.4); тут можна ствердити, що операції напряму є контекстно-залежними [6].

{Видалення на поз. 7} → {Видалення на поз. 12}, (1)

{Вставка '9' на поз. 7} → {Вставка '9' на поз. 12}, (2)

	Документ Клієнта 2												
Індекси	0	1	2	3	4	5	6	7	8	9	10	11	12
	Н	Т	У	У		К	П	І		1	8	9	8
Видалення на поз. 12	Н	Т	У	У		К	П	І		1	8	9	
Вставка '9' на поз. 12	Н	Т	У	У		К	П	І		1	8	9	9

Рисунок 2.4 – Виконання правильних трансформованих операцій Клієнта 1 Клієнтом 2

Розглянемо ще один більш наглядний та формальний приклад (рис. 2.5).

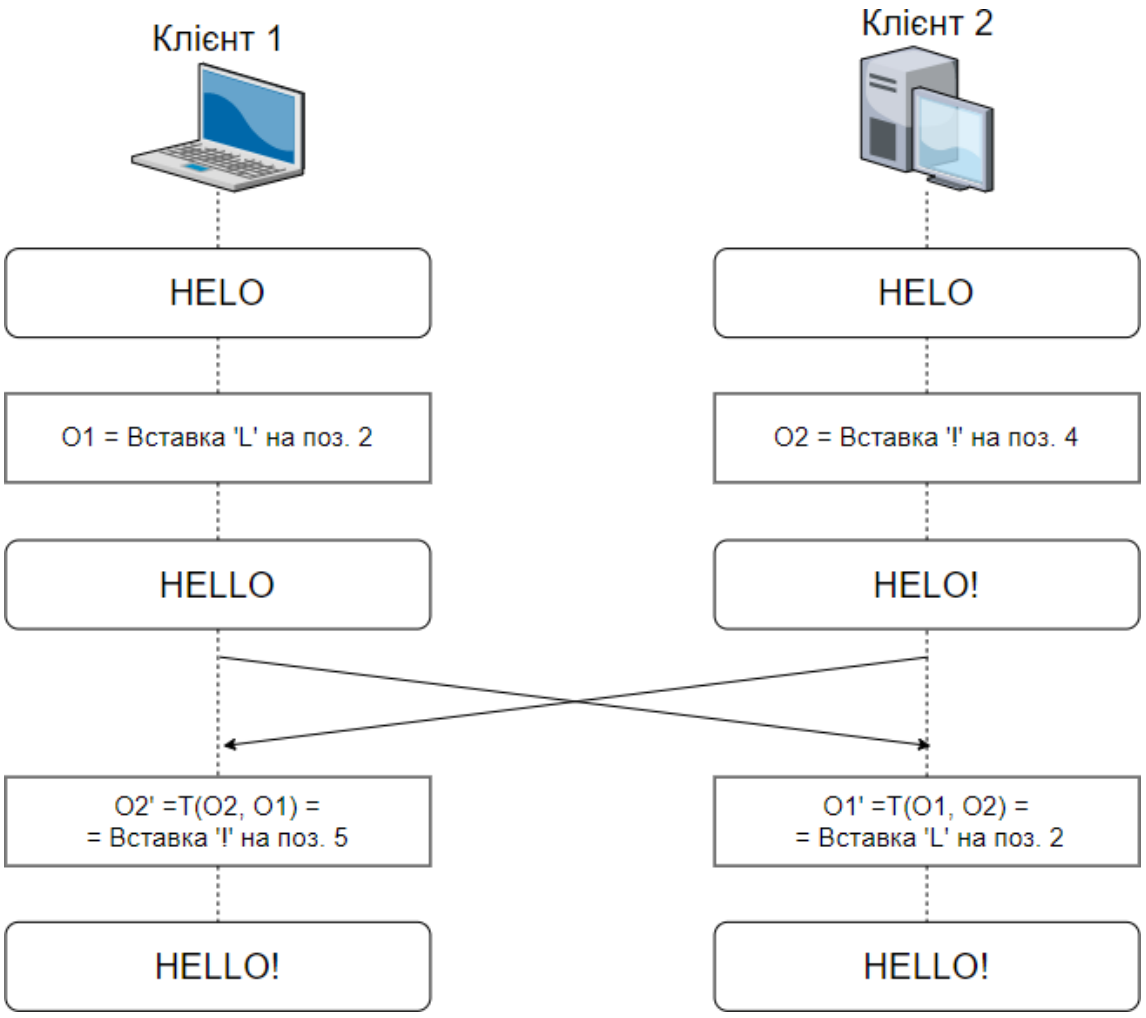

Рисунок 2.5 – Виконання трансформованих операцій

Початковим синхронізованим станом документу для обох клієнтів є “HELO”. Клієнт 1 виконав операцію O1 вставки ‘L’ на позиції 2, в той час як Клієнт 2 - операцію O2 вставки ‘!’ на позиції 4. Потім відбувається трансформація наступним чином T(Отримана операція, Локальна операція) до отриманих від інших клієнтів операцій. На виході було отримано коректну фінальну версію у обох клієнтів – “HELLO!” [7].

2.2. Особливості моделей операціональних трансформацій

Майже усі системи, що будуються на основі операціональних трансформацій, притримуються стандартів використання певних моделей узгодженості, які були створені розробниками технології OT. Існує 4 основних моделі, які взаємодоповнюють одна одну – модель CC, CCI, CSM та CA [8]. Кожна модель характеризує кілька властивостей узгодженості використання операціональних трансформацій. Коротко розглянемо кожну модель та її особливості.

CC модель визначає 2 головні стандарти управління узгодженістю для систем сумісного редагування. Це Causality Preservation (Збереження Каузальності) та Convergence (Збіжність) [9]. Перша властивість досить складна для розуміння і передбачає, що порядок виконання залежних операцій буде таким же як і причинно-наслідковий порядок, згідно процесу сумісної роботи. Друга властивість гарантує факт того, що після виконання всіх операцій над даними на всіх комп’ютерах користувачів буде відтворена одна й та сама версія даних [10].

Друга модель узгодженості називається CCI та містить всі властивості моделі CC, а також одну важливу додаткову властивість, яка зветься Intention Preservation (тобто збереження намірів). Вона передбачає, що операція, яку ми хочемо виконати над певними даними, має той самий зміст, який в ній і закладений. Говорячи простими словами, відбувається збереження намірів

користувача щодо зміни стану даних (після виконання операції поточний стан даних буде таким же як і передбачав сам користувач) [11]. Варто додати, що CCI модель узгодженості не розроблялась з метою перевірки пра вильності методів, які використовуються в алгоритмах операціональних трансформацій. Дана модель є незалежною від типів даних та операціональних моделей для конкретного додатку. Вона лише передбачає виконання протоколу властивостей, які входять до її складу [12].

Наступною моделлю узгодженості є CSM (Causality, Single-operation effects, Multi-operation effects) [13]. Перша її властивість аналогічна моделі CC. Друга і третя властивості подібні за своєю логікою; різниця між ними полягає лише в кількості застосованих до даних операцій. Сама ж властивість гарантує збереження ефекту впливу операції для будь-якого стану даних, а також залежність ефектів операцій у разі якщо їх було декілька.

Останньою моделлю є CA модель узгодженості, яка з'явилась через те, що CSM модель вимагає, щоб система мала загальний порядок всіх об'єктів[14]. Крім властивості каузальності, дана модель містить властивість допустимості, яка свідчить про те, що виклик кожної операції роботи з даними є допустимим для поточного стану даних та гарантує те, що застосування операції не порушить будь-які зв'язки між даними, а тим більше їх цілісність та саму структуру[15].

На основі цих моделей були створені алгоритми управління розподіленими системами з використанням операціональних трансформацій [16].

2.3 Загальні відомості про технологію Global – OT

Global – OT – це високорозвинена технологія для розробки послідовно розподілених сховищ даних, які поєднані одне з одним, використовуючи мережу Інтернет, семантику транзакцій та автоматичне вирішення

конфліктних ситуацій, використовуючи алгоритми операціональних трансформацій.

Технологія Global - OT надає розробникам доволі великий інструментарій з широким набором специфікацій та реалізацій для створення додатків з використанням принципів, які концептуально нагадують таку доволі поширену систему як Git, але є більш просунутою та багатоцільовою.

Головними особливостями даної технології є:

- збереження даних та високий рівень безпеки даних;
- ефективне проєктування мережі;
- робота з будь-яким, визначеним користувачем, типом даних.

Високий рівень безпеки даних пояснюється тим, що:

- всі дані зберігаються у вигляді фіксованих графів і зберіга ються в репозиторіях. Доступ до них відбувається по імені та з використанням публічного ключа. Варто зазначити, що в програмному забезпеченні репозиторій – це спеціальний сервер, на якому зазвичай зберігають різного роду дані – від звичайних масивів даних до програмного забезпечення різноготипу.
- Репозиторії надійно захищені від можливих сторонніх модифікацій:
  - лише власник приватного ключа може робити зміни даних у сховищі;
  - процес аутентифікації проводиться шляхом перевірки відкритим ключа до репозиторію;
  - за бажанням користувача проведені зміни в репозиторіях також можна зашифрувати;
  - всі внесені зміни в репозиторіях є адресо-залежними, що робить історію всіх внесених змін у графах незмінною.

Ефективне проєктування мережі пояснюється тим, що:

● Global - OT має повністю децентралізовану багаторівневу структуру мережі. Тому немає необхідності у централізованої служби для підтримки інформації про конфігурацію, іменування, забезпечення розподіленої синхронізації і надання групових служб.

● Сервери Global - OT можуть бути розташовані безпосередньо на невеликій відстані від користувачів (в тому ж самому дата-центрі, в мережі того ж самого провайдера, що і користувач, або ж навіть цей сервер може бути напряму вбудований в додаток). Це означає дуже низький рівень затримки передачі інформації, а також високий рівень доступності до даних.

● Так як історія внесених змін у графи є незмінною, це дозволяє технології Global – OT кешувати та робити попереднє завантаження даних, тобто:

- користувачі можуть підключитися до будь-яких серверів Global-OT, які можуть бути розташовані по всьому світу для отримання кешованих даних або попередньо завантажених змін будь-якого репозиторію;
- мережа навантажується поступово – чим більше відповідних репозиторіїв є доступними, тим більше створено їхніх кешованих копій.

● Конструкція мережі такого типу забезпечує високу відмовостійкість та дозволяє обробляти одночасне здійснення різних змін від найрізноманітніших джерел.

● Всі сервери Global-OT є взаємозамінним та агностичними до додатків, тобто будь-який сервер Global-OT може працювати з будь-якими додатками або репозиторіями Global-OT.

Робота з будь-яким, визначеним користувачем, типом даних та з графами будь-якого рівня складності.

● Global-OT може працювати з будь-якими визначеними користувачем типами даних. Ці дані зберігаються у вершинах графів. Ребра цих графів зберігають визначені користувачем операції, які можна застосувати до даних.

● Global-OT використовує визначені користувачем правила OT.

● Система Global-OT автоматично об'єднує графи будь-якої складності, за допомогою спеціального алгоритму рекурсивного злиття. Він розділяє складний граф на атомарні операції операціональних трансформацій (рис. 2.6)

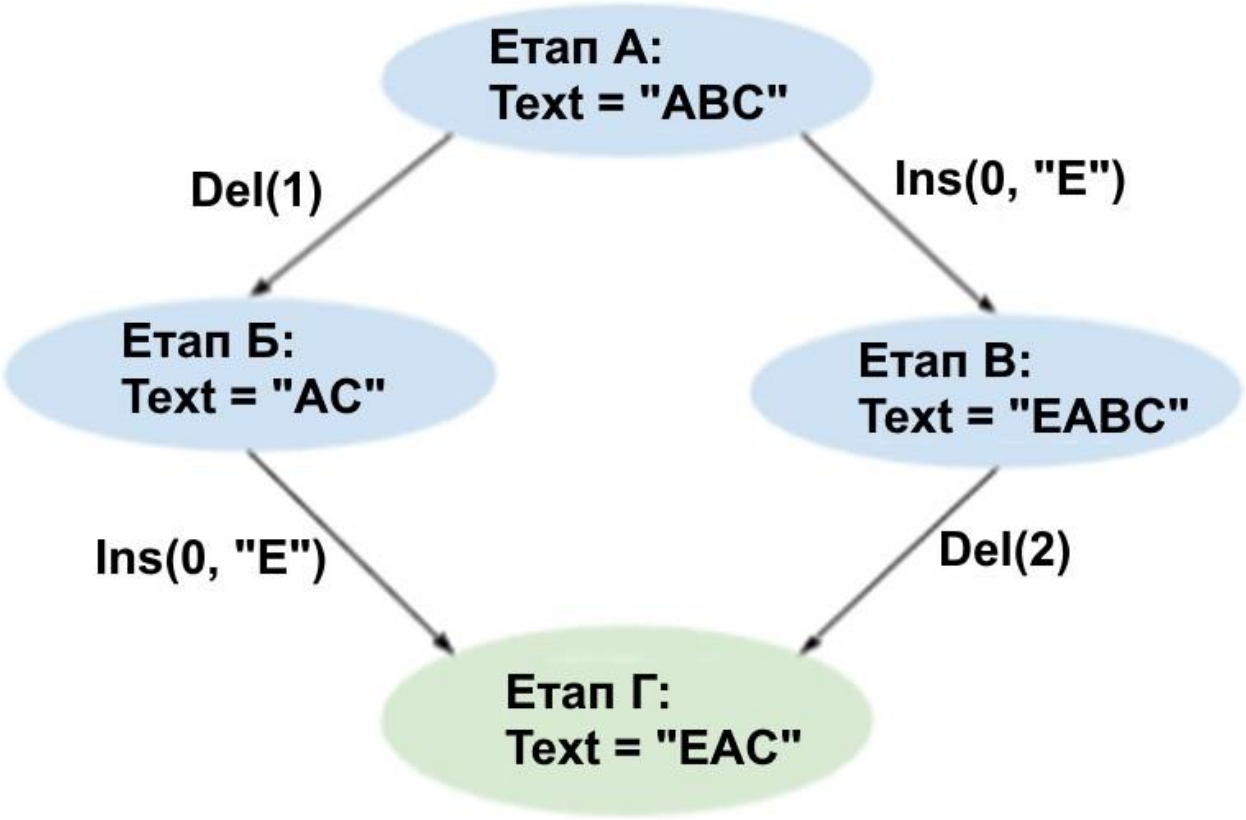

Рисунок 2.6 – Результат був би некоректним, якщо б операція видалення була б просто продубльована під час обробки змін на етапах B та C в етап D

2.4 Структури зберігання даних в Global-OT

Структура репозиторію в Global-OT представляється у вигляді графів різної складності. Вершини графів представляють собою дані, які заносить користувач, а ребра графів представляють собою перелік операцій, визначених користувачем, які застосовуються почергово до кожного значення батьківської вершини для створення наступних дочірніх вершин графа.

Розглянемо приклад (рис. 2.7).

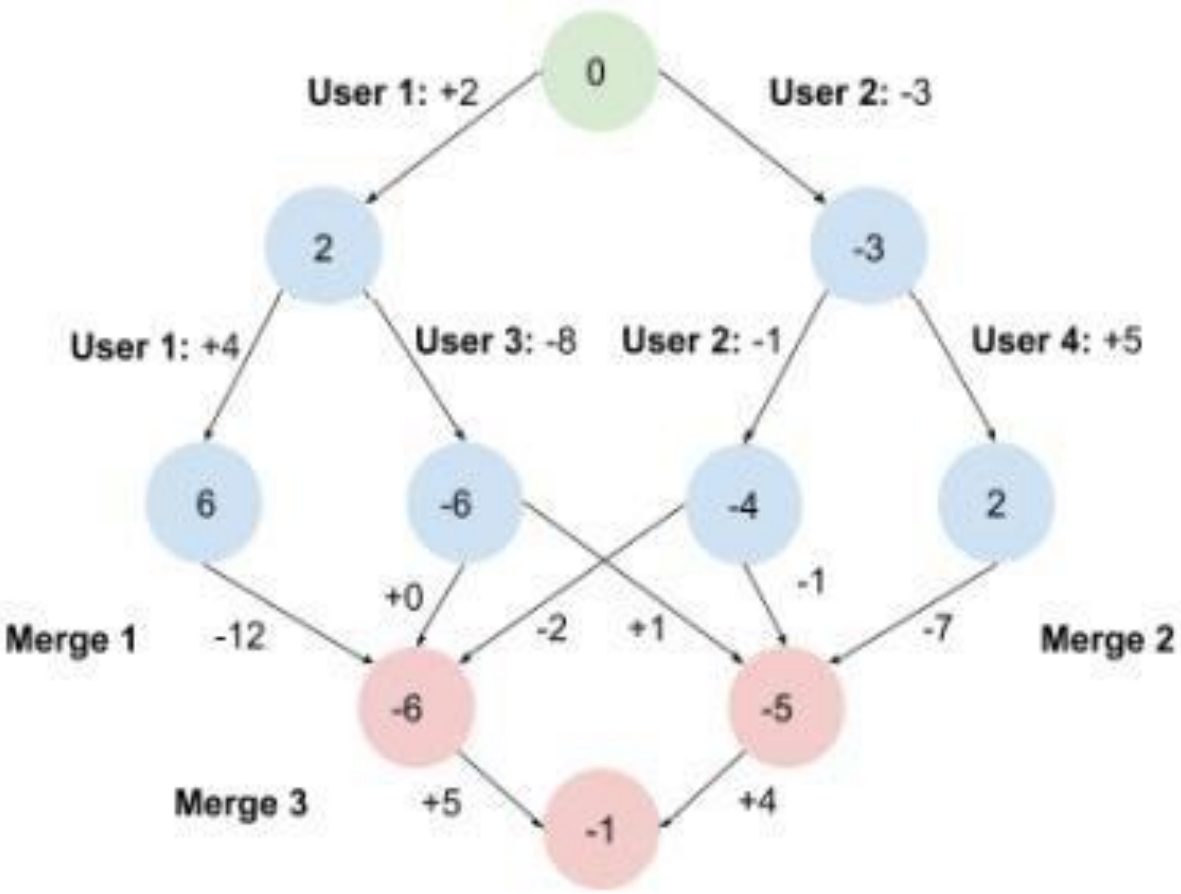

Рисунок 2.7 – Приклад зберігання інформації та списку операцій у Global-OT

Зеленим кольором на рис. 2.7 зображено стартову вершину, корінь графа. Фактично це стартовий стан і його значення дорівнює нулю.

Голубим кольором зображено проміжні вершини, стани яких відображено після виконання змін здійснених кожним користувачем.

Червоним кольором відображено вершини, стани яких згенеровано автоматично з використанням технології Global-OT, а саме за допомогою алгоритму злиття. Кінцевим результатом всіх операцій, які виконували користувачі є -1.

Варто зазначити, що для кожного користувача відображується своя версія репозиторію, яка відрізняється від репозиторіїв інших користувачів. Це пояснюється наступними особливостями.

- У кожного користувача є своя індивідуальна історія внесених змін та операцій над репозиторієм, яка потім буде послідовно синхронізована з іншими користувачами.
- Кожний етап синхронізації або ж злиття (merge) включає в себе різну кількість вершин всередині підмножини послідовних змін. Цей процес злиття відбувається послідовно до моменту отримання остаточного результату.

Наприклад, на рис. 2.7 проілюстровано, що перед злиттям Merge 1 не було синхронізовано зміни з користувача User 4 (так само як і користувачів User 1, User 2, User 3). А перед злиттям Merge 2 ну було синхронізовано зміни User 1 (аналогічно як і в попередньому випадку й не було занесено зміни User 2, User 3, User 4). Але врешті-решт було отримано фінальне злиття Merge 3 та отримано стійкий та правильний результат. Після цього всі користувачі отримали однакову історію внесених змін.

Для оптимізації продуктивності роботи технологія Global-OT може зберігати деякі проміжні результати у вершинах графів. Таким чином, Global-OT позбавляється необхідності у виконанні передачі всього графу та

виконувати застосування всіх змін від самого кореня до обчислення потрібного користувачеві вузла.

Кожен репозиторій має свій індивідуальний набір ключів – публічний та приватний ключі.

Публічний ключ – це унікальний ідентифікатор кожного репозиторію. Користувачі можуть виконувати пошук потрібних для них репозиторіїв, використовуючи їх. Також варто зазначити, що публічні та приватні ключі є пов'язаними між собою.

Також варто зазначити, що існує декілька вимог до OT операцій, яких дотримується Global-OT.

- Операції OT повинні бути зворотними. Якщо застосувати певну довільну операцію до стану, а потім застосувати інверсію до цієї операції, повинно бути отримано початковий стан (рис. 2.8, а).
- OT операції повинні мати властивість склеювання. Припустимо, користувач застосував певну послідовність операцій для отримання певного результату, і якщо користувач виконає склеювання цієї послідовності операцій, він повинен отримати однаковий результат (рис. 2.8, б).
- Склеювання будь-якої операції та її інверсії не будуть спричиняти жодних дій, або ж так звану no-op операцію (рис. 2.8, в).

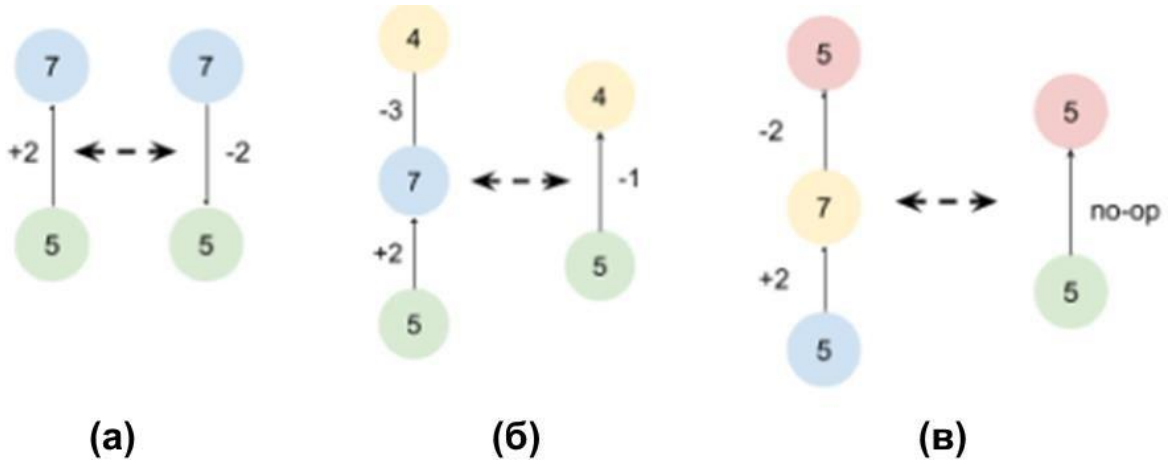

Рисунок 2.8 – Демонстрація вимог до OT операцій, яких дотримується Global-OT

- При склеюванні будь-якої операції та no-op операції буде отримано оригінальну операцію (рис. 2.9, г).
- При виборі будь-якого маршруту проходження від одного стану (наприклад, початковий стан в корені графа) до іншого стану (наприклад, поточний стан), включаючи переходи назад із інверсіями користувач повинен отримати завжди один й той самий результат (рис. 2.9, д).

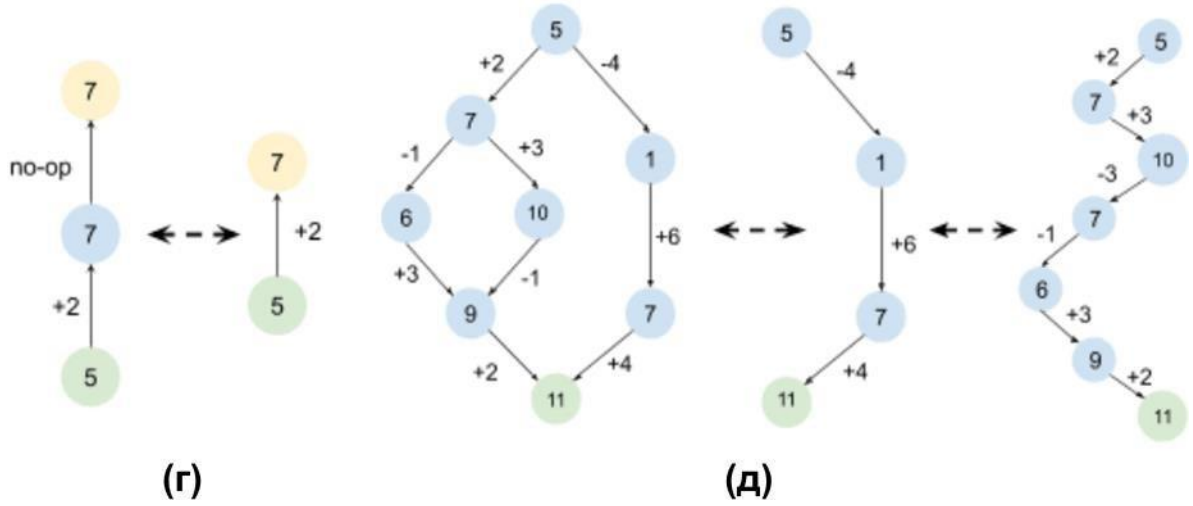

Рисунок 2.9 – Демонстрація вимог до OT операцій, яких дотримується Global-OT (продовження)

В загальному випадку найширшого застосування в розробці програмного забезпечення набув такий формат даних, як JSON, але цей формат абсолютно не підходить для зберігання графів. Так як граф необхідно передавати в бітовому представленні, використовується такий формат даних, як BLOB.

З ОТ серверу надсилаються коміти, які зберігаються в форматі BLOB. Тому, клієнтська частина має містити два серіалайзери (спеціальні перетворювачі даних). Вони можуть працювати з даними як формату JSON, так і з даними BLOB типу.

Тепер варто розглянути структуру кожної зміни, або ж коміту (від англ. commit).

- Кожна зміна автоматично підписується власником приватного ключа (таким чином, лише власник репозиторію може робити коміти).
- Кожен коміт може бути додатково зашифровано за допомогою самозгенерованого симетричного ключа використовуючи алгоритм AES.
- Лише кінцеві користувачі, які мають симетричний ключ, можуть розшифровувати коміти.
- Підписи можуть бути перевірені будь-яким проміжним сервером Global-OT, з яким працює сховище, навіть якщо воно не забезпечено ключем шифрування. Це дозволяє перевірити справжність комітетів для безпеки кешування та попереднього завантаження.
- Кожен коміт містить цілий перелік спеціальної службової інформації: ідентифікатор репозиторію, часову мітку, тобто час його створення, ідентифікатор батьківського коміту, а також весь перелік операцій, що було застосовано до поточного стану даних (рис. 2.10).

Рисунок 2.10 – Узагальнена структура коміту в Global-OT

2.5. Висновки

У даному розділі розглянуто всі основні властивості та наочно продемонстровано переваги використання операціональних трансформацій у розробці сучасного програмного забезпечення.

3. ОПИС РОБОТИ МОДУЛІВ СЕРВЕРНОЇ ЧАСТИНИ ВЕБДОДАТКУ

3.1. Застосування операціональних трансформацій на серверній частині вебдодатку з використанням Global-OT

Основні особливості та переваги операціональних трансформацій розкриваються саме на серверній частині вебдодатку з використанням Global-OT.

Global-OT – це технологія, яка включає в себе багаторівневу мережу з клієнтським рівнем, P2P простором OT - серверів та декількома незалежними центрами управління (Discovery Services) (рис. 3.1).

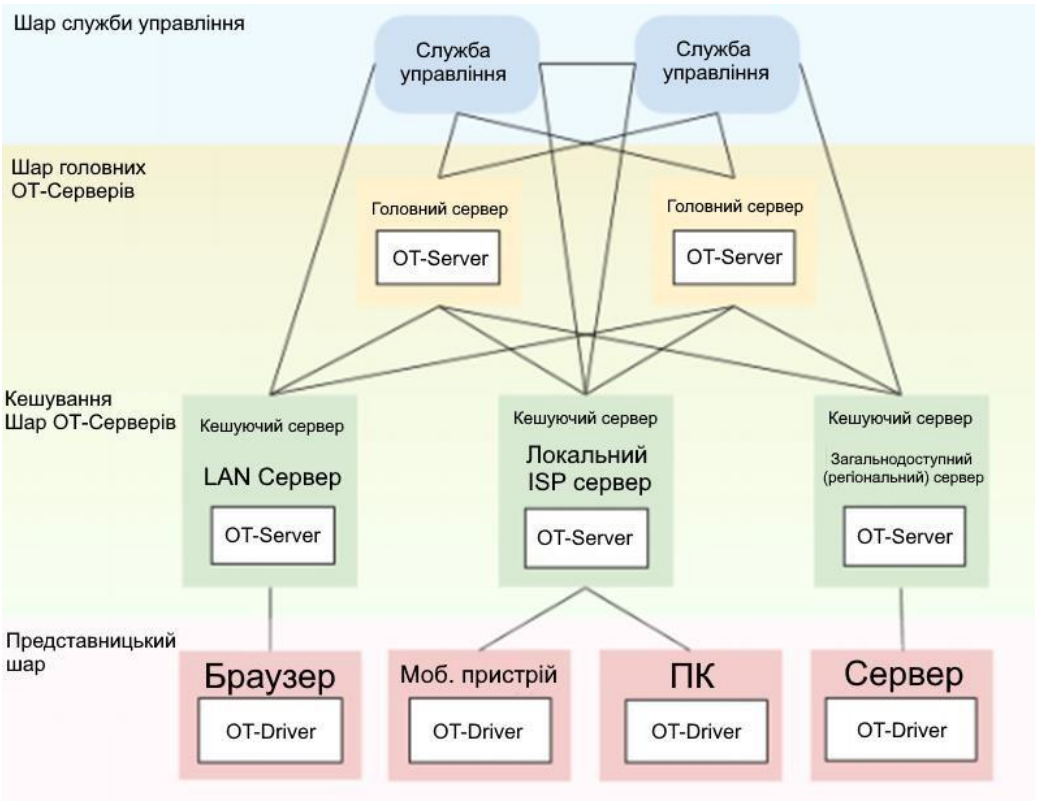

Рисунок 3.1 – Приклад мережі технології OT

Розглянемо особливості структурних компонентів мережі операціональних трансформацій.

OT-Driver забезпечує взаємодію між клієнтською програмою та OT-сервером, тобто:

- створює репозиторії та видаляє їх з головних серверів (Master servers);
- вносить зміни (коміти) до репозиторіїв користувачів (завантажує та об'єднує всі зміни від віддалених репозиторіїв до локальних);
- зберігає зміни (коміти) до репозиторіїв (вивантажує зміни від локальних репозиторіїв до локальних);
- відправляє запити вилучення (відправка запропонованих користувачем дій в системі стороннім особам; зміни можуть бути або ж прийняті власником репозиторію або ж відхилені).

Варто зазначити, що кожен OT-Driver не може зв'язуватися з іншим OT-Driver напряму.

OT-Server може бути головним (Master) за умови, якщо він є оригіналом репозиторію, а також може бути Caching OT-Server, якщо ж він зберігає (іншими ж словами кешує) репозиторій. Також OT-Server може комбінувати обидві ролі, надаючи послуги кешування, а також приймати та завантажувати зміни користувачів до репозиторію.

Користувачі самі вирішують, до якого OT-сервера вони хочуть підключитися, для завантажити чи вивантаження репозиторію. Зазвичай використовується один з наступних OT-серверів:

- LAN сервер;
- локальний ISP сервер;
- загальнодоступний сервер, який фізично розташований на відносно невеликій відстані.

Також варто зазначити, що ролі серверу не мають будь-яких привілеїв, тобто вони мають однаковий пріоритет операцій.

Розглянемо основні завдання центру управління (Discovery Service).

● Зберігає інформацію про IP-адреси головних (Master) серверів, а також публічні ключі репозиторіїв, які зберігаються на головних серверах, на кшталт DNS.

● Пропонує найбільш близько розташований головний сервер для завантаження та вивантаження даних.

● Перевіряє чинність, тобто проводить процес валідації, підпису пакетів повідомлень.

Розглянемо основні особливості кешування.

● Щоразу, коли новий репозиторій проходить через кешуючий OT-сервер, кеш репозиторію зберігається на сервері (якщо функція кешування ввімкнена).

● Якщо користувач спробує отримати доступ до того ж репозиторію наступного разу, серверу більше не потрібно буде підключатися до центру управління (Discovery Service).

● Кешування автоматично реалізується в додатках Global-OT. Як результат, вони мають високу швидкість доступу до даних та низьке навантаження на пропускну спроможність.

Типовий процес завантаження репозиторію зображено на рис. 3.2.

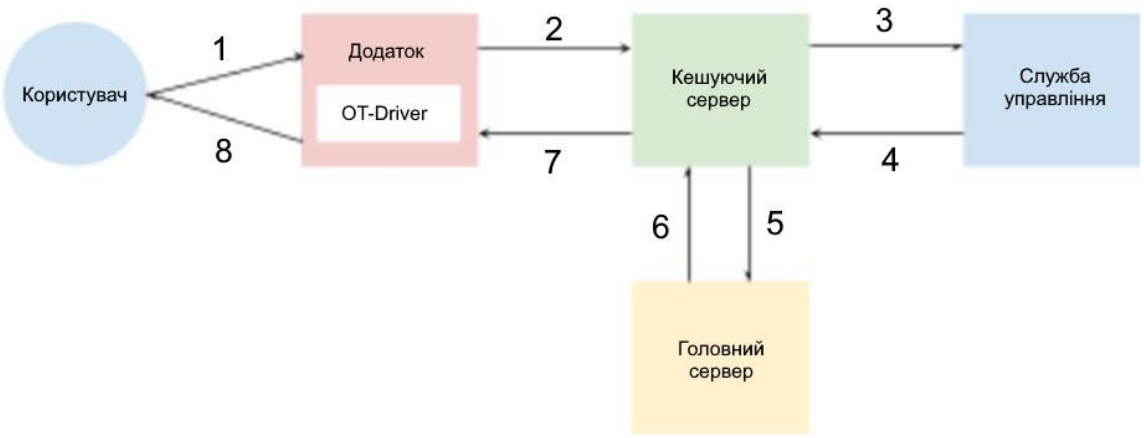

Рисунок 3.2 – Процес завантаження репозиторію

Повторний процес завантаження репозиторію зображено на рис.3.3.

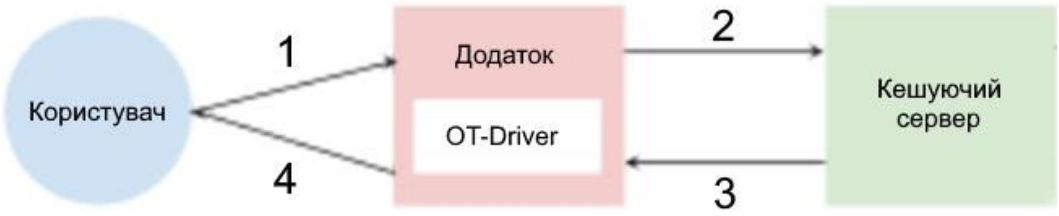

Рисунок 3.3 – Повторний процес завантаження репозиторію

Безпека кешування забезпечується двома важливими властивостями Global-OT.

- Графи комітів сховищ є незмінними. Це гарантує, що всі сервери зберігають однакову інформацію.
- Цифровий підпис на кожному зобов'язанні гарантує, що вони не були змінені або підроблені сторонніми сторонами.

Кешування також забезпечує попереднє завантаження (prefetching). Ті репозиторії, доступ до яких потребується найчастіше автоматично кешуються на OT-серверах, збільшуючи швидкість доступу до даних.

3.2 Вибір інструментарію для розробки програмного забезпечення

Вибір інструментарію для розробки програмного забезпечення є доволі кропітким питанням. Зараз є дуже широкий набір схожих за своєю роботою інструментів для розробки різного виду програмного забезпечення, проте функціонал може різнитися від одного продукту до іншого.

Так як технологія Global-OT була написана написана мовою програмування Java, то для повноцінного та успішного користування цією технологією, мовою нашого проєкту також було вибрано мову Java.

На сьогоднішній момент мова Java є однією з найпоширеніших та найпопулярніших мов програмування. Перша версія мови з'явилася ще в 1996 році й розроблялася як універсальна мова програмування, яку можна застосовувати для розробки різного роду завдань. Станом на сьогоднішній день останньою версією є Java 14, реліз якої відбувся в березні 2020 року, але найбільш широкого поширення набула версія Java 8, на якій і написано додаток, так як саме ця версія забезпечує найширшу підтримку найрізноманітніших платформ без виникнення помилок.

На даний момент найпопулярнішим середовищем розробки для Java є IntelliJ IDEA від компанії JetBrains (по ряду опитувань станом на 2019 нею користувалося близько 60% розробників на Java) [17]. Саме тому вибрано саме це середовище розробки (рис. 3.4).

Рисунок 3.4 – Середовище розробки IntelliJ IDEA

При розробці клієнтської частини вебдодатку використано мову Javascript з бібліотекою React.js. Ця бібліотека є дуже сучасною та надає широкий спектр можливостей для створення інтерфейсів користувача. Також

вона вирішує цілий ряд проблем з вирішенням проблем, які пов'язані з оновленням інформації на вебсторінці, а виконує зберігання даних.

Для написання клієнтської частини вебдодатку використано середовище розробки WebStorm від компанії JetBrains (рис 3.5).

Рисунок 3.5 – Середовище розробки WebStorm

Обидва середовища розробки від компанії JetBrains мають безкоштовні навчальні версії, доступ до яких надається після підтвердження того, що користувач на даний момент є студентом. IntelliJ IDEA та WebStorm має цілий перелік схожих та корисних функцій – автозавершення тексту деяких команд, які вносить розробник, повністю зображує синтаксис використаної мови програмування, надає можливість збереження різних версій тексту програми на різні сервіси, такі як Github [18].

Також IntelliJ IDEA та WebStorm мають гарні можливості відлагодження коду – є можливість знаходженння помилкових ділянок коду, які спричиняють проблеми в компіляції програми, а також є можливість виконання відлагодження коду програми в ручному режимі шляхом встановлення

спеціальних точок (breakpoint), до яких буде виконано код програми з майбутнім покроковим тестуванням програми.

Також в проєкті використовується система для автоматизації роботи вебдодатку – Docker. Він надає можливість «запаковувати» додатки зі всім допоміжним програмним забезпеченням в спеціальні контейнери, які можна переносити на будь-які сучасні ОС.

Для автоматизації роботи проєкту, а також для запуску проєкту на локальному сервері використано фреймворк Maven. В першу чергу він використовується для управління та зпакування програми у зручний для користувача вигляд.

3.3 Опис структурних модулів серверної частини вебдодатку

Серверна частина вебдодатку складається з декількох різних модулів, які в поєднанні між собою складають цілісну систему для роботи з технологією операціональних трансформацій.

Розглянемо ці модулі.

- Модуль OTAlgorithms.java, який містить в собі складні алгоритми операціональних трансформацій, які використовуються для коректної роботи з графом операціональних трансформацій.
- Модуль OTRepository.java, який фактично являється сховищем графу для зберігання інформації конкретного користувача.
- Модуль MergedOTSystem.java відповідає за злиття операцій внесених різними користувачами, ґрунтуючись на основних властивостях операціональних трансформацій. Фактично цей модуль відповідає за всі основні математичні обрахунки в додатку. Варто зазначити, що клієнтська частина вебдодатку має модуль, який виконує аналогічні

операції над всіма даними. Він є ключовим модулем в клієнтській частині вебдодатку.

- Модуль GraphReducer.java містить набір методів, які дозволяють коректно працювати з графом та формувати результат даних у вигляді комітів, які будуть додані в сам граф.
- Модуль OTDataManager.java відповідає за збереження даних в вищеозгаданому бінарному форматі BLOB.
- Модуль Routing.java відповідає за маршрутизацію між клієнтом (вебдодатком) та сервером, тобто він виконує зв'язуючу функцію між цими двома ланками.
- Модуль OTException.java відповідає за обробку помилкових ситуацій, які можуть виникнути при роботі всіх модулів в серверній частині вебдодатку.
- Модуль Configuration.java приймає набір конфігураційних файлів для коректної роботи вебдодатку та зберігає приватні та публічні ключі в зашифрованому вигляді.

Розглянемо будову вищезазначених модулів детальніше.

Основними методами в реалізації модуля OTAlgorithms.java є методи Merge, CheckOut, LoadGraph, Copy та GetCommit.

Метод Merge виконує послідовне злиття внесених змін різними користувачами. Цей метод також буде викликано за умови, якщо в якогось користувача було нестабільне підключення до мережі Інтернет. Тобто цим методом буде внесено всі зміни, які робив користувач, поки підключення було відсутнє.

Метод CheckOut відповідає за ініціалізацію початкового стану графу, в який заносяться всі внесені користувачами зміни. Зазвичай цей метод викликається з самого початку роботи додатку. Тобто коли користувач запускає додаток, цей метод CheckOut буде викликано автоматично для

отримання поточної версії віддаленого серверного графа операціональних трансформацій.

Метод LoadGraph при наявності змін даних від користувача може додавати окремий коміт із цими даними до репозиторію користувача та повертати видозмінений на основі цього репозиторію.

Метод Copy відповідає за отримання поточних станів даних обох користувачів й виконує взаємне копіювання один між одним. Для цього йому необхідно отримати дані про батьківські, тобто попередні стани репозиторію. Якщо користувач має кілька змін (комітів), то відбувається послідовне копіювання, а всі дані стають в чергу згідно пріоритетності застосованих операцій та часу їх виконання.

Метод GetCommit виконує отримання внесених змін до репозиторію. Цей метод використовується в багатьох модулях, він повертає унікальний номер коміту, час внесених змін, а також сам вміст цього коміту.

Основними методами в реалізації модуля OTRepository.java є методи GetLevel, GetHead та PushingUpdateHead.

Метод GetLevel використовується для безпосередньої роботи з графом, тобто для отримання рівня дерева графу, на якому розташований потрібний коміт, який необхідно передати.

Метод GetHead використовується для отримання найактуальнішої версії даних, тобто для отримання останньої ревізії внесених змін, що містяться на сервері.

Метод PushingUpdateHead використовується для надсилання найактуальнішої версії даних, тобто для надсилання останньої ревізії внесених змін, на сервер.

Основними методами в реалізації модуля MergedOTSystem.java є методи MergeOTSystem, Transform, Squash, Invert та isEmpty.

Метод MergeOTSystem може приймати довільну кількість операцій. Він може визначати пріоритетність операцій та надає можливість системі

операціональних трансформацій (OT-System) послідовно застосовувати до них основні властивості операціональних трансформацій.

Метод Transform виконує перетворення всіх операцій, чим покращує загальну швидкодію.

Метод Squash відповідає за об'єднання необхідних для обчислення операцій з метою підвищення загальної швидкодії.

Метод Invert відповідає за виконання обернених операцій. Він може виконувати інверсію не тільки одиночних операцій, а й цілого списку різнотипних операцій.

Метод isEmpty виконує перевірку операції на її пустоту. Пустою вважається така операція, після застосування якої до даних поточний стан даних не зміниться. Перевірку пустої операції необхідно робити як тільки на сервер надійшла певна операція, оскільки якщо в результаті перевірки операція виявилась пустою, то не потрібно робити додаткових перетворень з операцією, а тим більше застосовувати її до даних. Даний метод значно спрощує роботу складної серверної системи операціональних трансформацій та оптимізує процес серверної обробки даних. Також, варто зазначити, що метод isEmpty викликається після виконання трансформацій або об'єднання кількох операцій, оскільки в процесі перетворень результуюча операція може виявитись пустою. В такому випадку дані не будуть потребувати її застосування.

Одним з основних модулів серверної частини будь-якого вебдодатку, розробленого на основі технології операціональних трансформацій є модуль, який повертає граф операціональних трансформацій (GraphReducer) з усією історію комітів. По своїй суті GraphReducer є інтерфейсом, який визначає метод ініціалізації графа з початковим набором комітів та клас-результат, який формується на основі всієї історії комітів. Даний модуль повертає результат у вигляді деревовидної структури, яку інші модулі вебдодатку можуть змінювати та використовувати для отримання інформації. Граф має

можливість додавання комітів даних до себе. Проте, додавання комітів в дерево-граф та збереження їх в історії графу можливо тільки за умови коректної структури комітів. В інакшому випадку спрацює модуль обробки помилок та поверне помилку-виключення про неправильний формат коміту. Граф можна розкладати на підграфи (рис. 3.6) та комбінувати коміти в ньому. Усі ці можливості задаються конфігураціями, переданими безпосередньо в клас формування результату в інтерфейсі графа. Як бачимо з рис. 3.6 базовий граф з результатом можна розділити на кілька дрібних графів, які по своїм властивостям повторюють базовий граф, проте містять менше інформації, або більше детально описують саме ту сукупність комітів даних, яка міститься в них .

Для збереження залежностей, які використовують підграфи базового графа використовується спеціальний Java-конфігуратор – Inject, який передає визначені залежності між модулями програми.

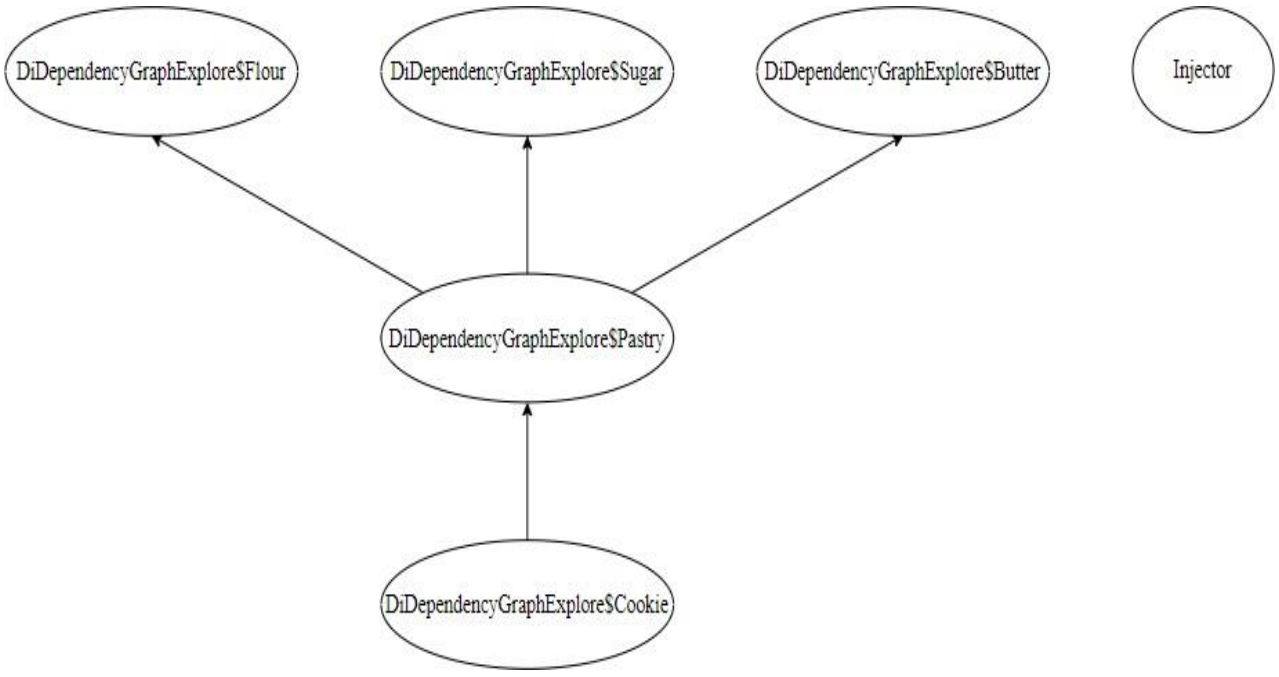

Рисунок 3.6 – Розділення базового графа на підграфи

Усі коміти, які йдуть від клієнтського програмного забезпечення операціональних трансформацій зберігаються модулем OTDataManager. Даний модуль включає набір методів для роботи з користувацькими даними та методи обробки серверних запитів до ОТ репозиторію. За своєю логікою методи подібні до команд в системі контролю версій Git.

Для взаємодії з віддаленим репозиторієм використовуються запити:

- push;
- rebase;
- fetch;
- poll;
- checkout;
- reset.

Даний модуль зберігає всі дані в зашифрованому вигляді і застосовує операції, що визначені користувачем, до даних завдяки методу apply. Основним методом в даному модулі є метод commit, завдяки роботі якого зберігаються всі внесені користувачами зміни до репозиторію.

Наступним важливим модулем серверної частини вебдодатку є модуль маршрутизації Routing. Він використовує конструкції Java servlet для маршрутизації між клієнтськими та серверними частинами додатку. Також він відповідає за кешування статичних файлів проєкту по заданому маршруту - /static/*. Він включає в себе маршрутизацію авторизації вебдодатку, саме завдяки цьому реалізовано коректну авторизацію користувачів за допомогою їх приватних ключів. Також модуль маршрутизації дозволяє отримувати дані на сервері завдяки визначеним серверним маршрутам:

- contacts/* - відповідає за отримання публічних ключів користувачів, які приймали участь у спільному редагуванні певного документа;
- documents/ - відповідає за отримання списку всіх документів користувача;

- document/:id/* - відповідає за отримання конкретного документу за його ідентифікатором;
- profile/:pubKey/* - відповідає за отримання профілю користувача за його публічним ключем;
- myProfile/* - відповідає за отримання публічного та приватних ключів користувача, який авторизувався в додатку.

Варто зазначити, що також є два важливих маршрути Routing/ot/* та Routing/debug/*, про які варто згадати в особливому порядку. Маршрут Routing/ot/* використовує спеціальний декоратор (він дозволяє динамічно додавати об'єкту нові властивості, не вдаючись при цьому до створення нових класів) сесій авторизації користувача в OT репозиторії та дозволяє отримати дані про поточну сесію користувача.

Маршрут Routing/debug/* використовується для тестування кожного з модулів серверної частини вебдодатку. При тестуванні модулів, що пов'язані з операціональними трансформаціями, надається можливість виведення графу репозиторію.

Модуль OTException відповідає за обробку всіх можливих помилок на серверній частині вебдодатку, а також за помилки, які можуть виникнути при виконанні операціональних трансформацій. У разі помилки призупиняє роботу серверної частини додатку, повідомляє про виникнення конкретної помилки й потім відновлює свою роботу. Під час виникнення помилки при роботі з OT репозиторієм будь-які нові зміни користувача не будуть внесені до репозиторію, при цьому граф комітів залишиться в попередньому стані. Також при спробі отримання коміту, який не міститься в OT репозиторії модуль повідомить користувача про його відсутність.

Модуль Configuration містить набір налаштувань для серверної частини вебдодатку. З його допомогою встановлюється порт запуску вебдодатку, вноситься приватний ключ в спеціальний файл на сервері. Відповідає за налаштування віддаленого сервісу керування (Discovery Service), а також

містить певні налаштування для роботи з OT репозиторієм. В цьому модулі міститься важливий конфігураційний файл global-documents.properties, в якому містяться всі налаштування для коректної роботи вебдодатку.

Взаємодію різних модулів програми з конфігураційним модулем зображено на рис. 3.7. Серверна частина використовує конфігураційний модуль для встановлення допустимої кількості з'єднань між вебдодатком та OT сервером (ThrottlingController). Такі з'єднання зберігаються у спеціальному класі WorkerPool, де кожне з'єднання має свій ідентифікатор WorkerId. Після надсилання запитів в OT репозиторій Eventloop виконує задані користувачем операції.

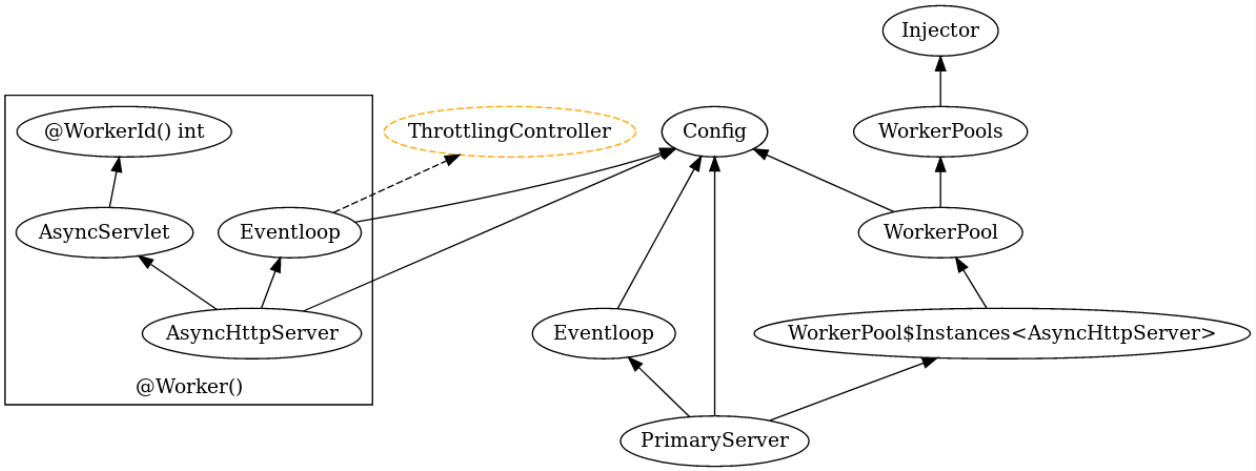

Рисунок 3.7 – Взаємодія інших модулів з конфігураційним модулем

Варто зазначити, що описано лише ключові методи, які містяться в кожному модулі.

3.4 Опис інтерфейсу вебдодатку

Для найкращої демонстрації роботи операціональних трансформацій вирішено розробити вебдодаток для спільного онлайн-редагування текстових документів. Для розробки даного додатку використано фреймворк DataKernel, написаний на мові програмування Java. Він містить передові технології для

швидкої та безпечної роботи над документами з використанням операціональних трансформацій на серверній частині вебдодатку. Основним цілями при створенні даного додатку були не тільки високопродуктивна робота, а й зручний та зрозумілий для користувача інтерфейс з достатньою кількістю необхідних для нього функцій в області редагування та збереження редагованих файлів.

При запуску вебдодатку користувачеві необхідно ввести приватний ключ, який він отримує від адміністратора додатку. Адміністратор генерує приватний ключ випадковим чином, посилаючи відповідний запит серверу. Початковий інтерфейс зображено на рис. 3.8.

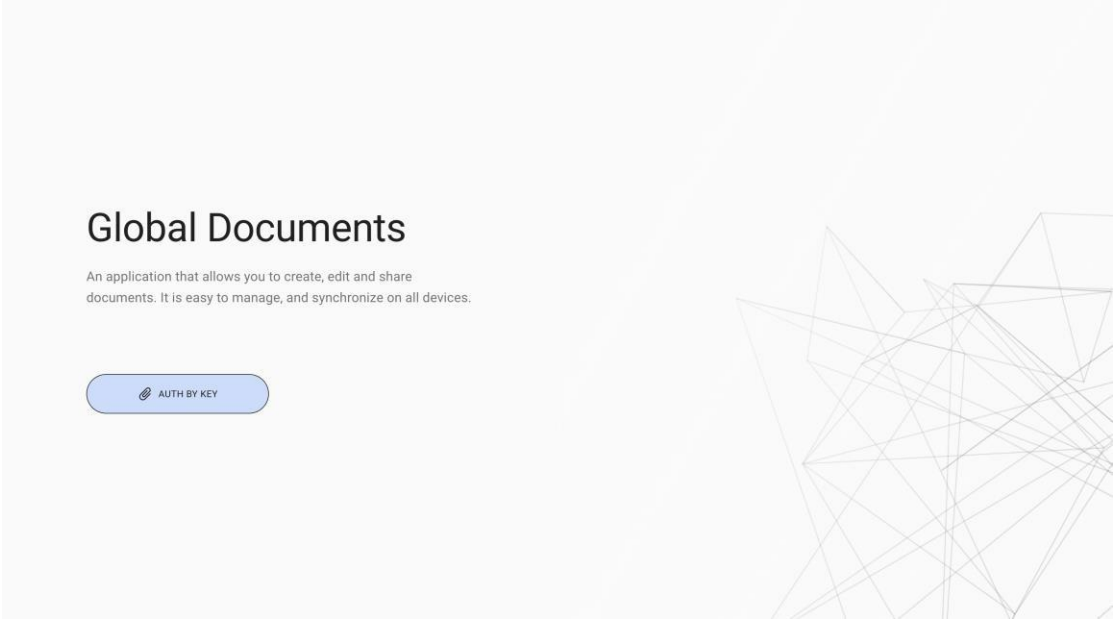

Рисунок 3.8 – Початковий інтерфейс розробленого додатку

Користувачі мають змогу зберігати та переглядати потім документи, в які вносилися зміни. Приклад перегляду списку документів користувача та зміст довільного документу зображено на рис. 3.9.

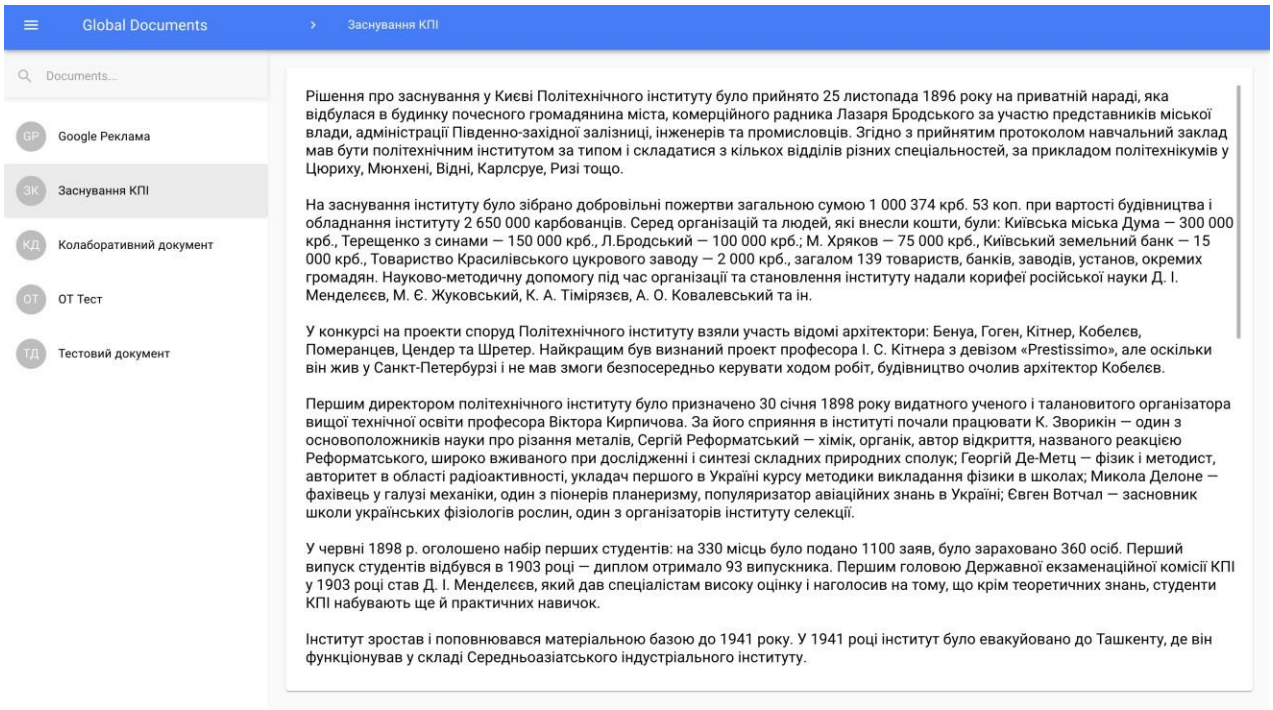

Рисунок 3.9 – Список документів користувача

Кожен користувач може надавати можливість до спільного редагування будь-якого документу іншим користувачам (рис 3.10). Надавати доступ можна користувачам, які вже є в контактному списку, або ж якщо користувач ще не був доданий до списку контактів його потрібно знайти та додати до цього списку використовуючи його індивідуальний публічний ключ. Цей ключ можна знайти в своєму профілі, де також є можливість зміни імені, що відображується для інших користувачів (рис 3.11).

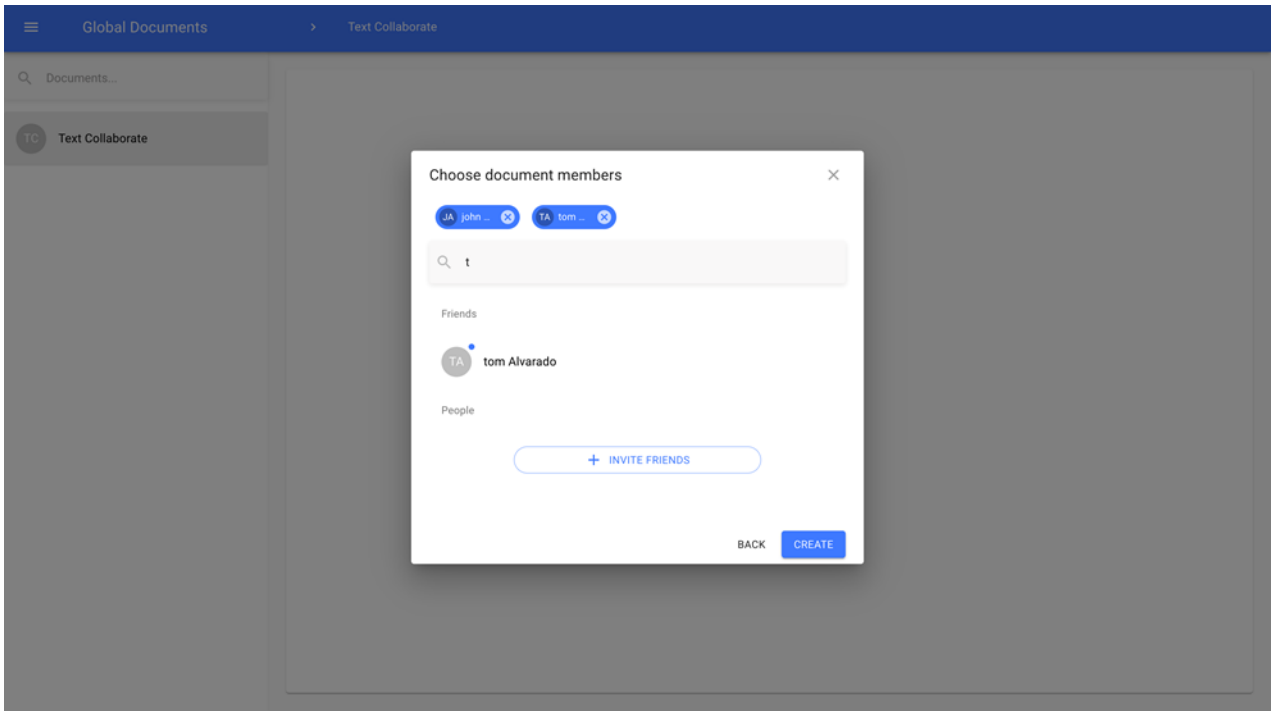

Рисунок 3.10 – Приклад надання можливості іншим користувачам до спільного редагування документу

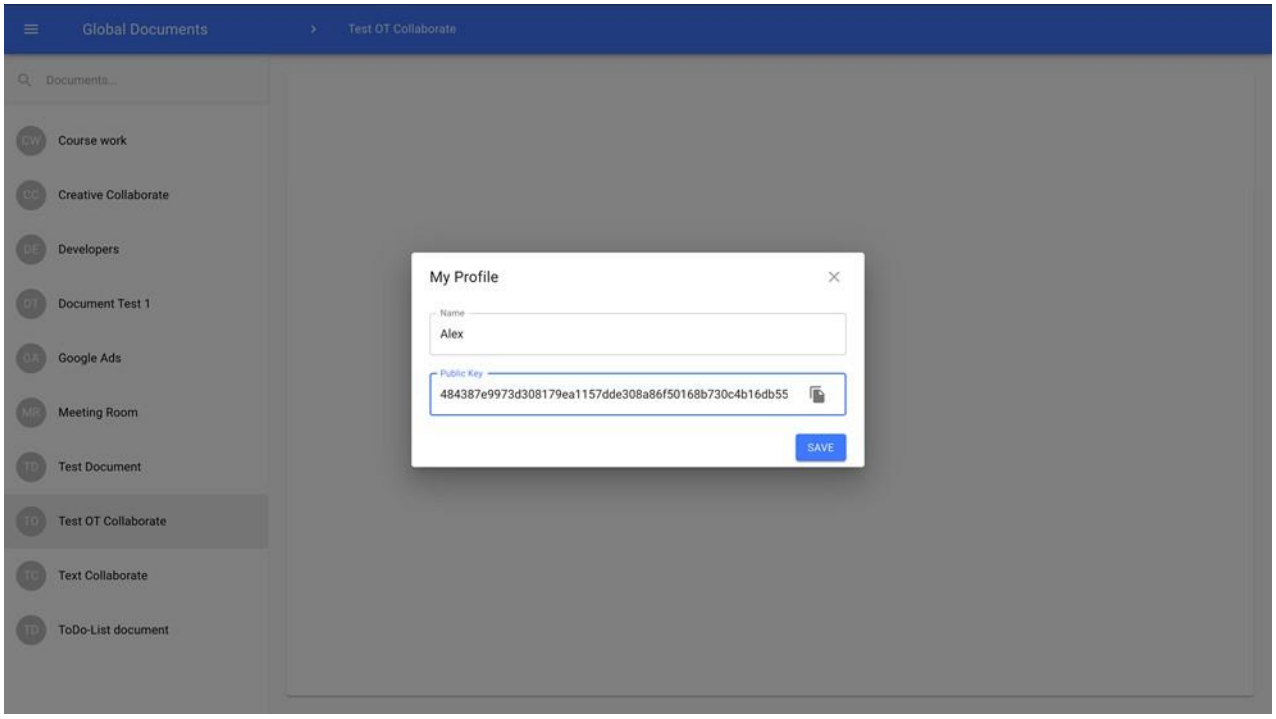

Рисунок 3.11 – Особистий профіль користувача та його публічний ключ

3.5 Аналіз опису роботимодулів вебдодатку

У даному розділі розглянуто загальну структуру розробленого програмного забезпечення – його модулі, методи та особливості реалізації вебдодатку з використанням операціональних трансформацій. Особливу увагу приділено опису серверної частини вебдодатку, адже саме там відбувається основна реалізація операціональних трансформацій розробленого текстового редактору. Також продемонстровано загальний приклад роботи додатку з відповідними настановами для правильної роботи користувача. Також проілюстровано тестування розробленого додатку та показано коректність виконання всіх операцій розробленого програмного забезпечення.

4 ТЕСТУВАННЯ СЕРВЕРНОЇ ЧАСТИНИ ВЕБДОДАТКУ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

4.1 Тестування програмного забезпечення

Під час розробки будь-якого програмного забезпечення важливим пунктом є відлагодження та тестування виконаної розробником роботи. Оскільки робота всього вебдодатку базується на операціональних трансформаціях, то важливо також не тільки перевірити правильність роботи самого додатку та його взаємодію з серверною частиною, а й переконатися в правильності виконання перетворень, які використовуються при роботі користувача з самим редактором тексту.

Коректність роботи операціональних трансформацій в текстовому редакторі зображено на рис. 4.1. Варто зазначити, що також на рисунку зображено повний граф комітів, який наочно відображує всю історію змін, внесених до документу. Як згадувалося раніше, раніше кожна зміна має свій унікальний ідентифікатор, який створюється шляхом генерування випадкової послідовності символів.

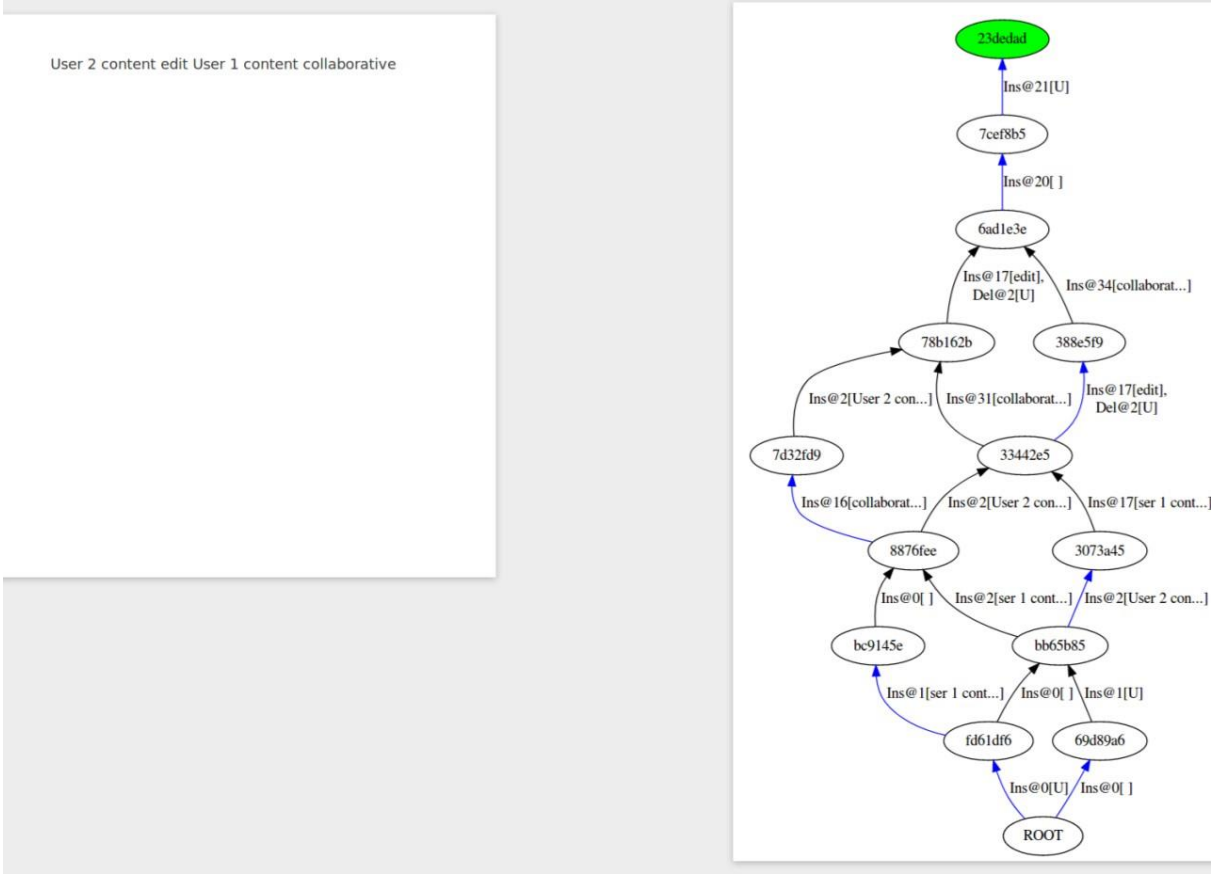

Рисунок 4.1 – Перевірка роботи редактору з наочною демонстрацією графу

На рис. 4.2 наочно продемонстровано приклад зберігання внесених змін двома різними користувачами у вигляді графу. Результатом, який буде збережено на сервер, буде слово “DOC”, адже User2 вніс більшу кількість актуальніших змін, не зважаючи на те, що роботу над документом до певного моменту продовжував User1.

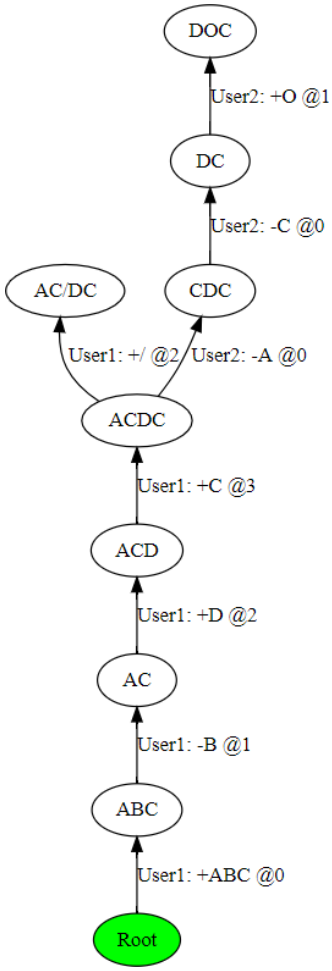

Рисунок 4.2 – Наочний приклад внесених змін у вигляді графу

Перевірити правильність взаємодії модулів серверної частини вебдодатку можна використовуючи спеціальний вебсервіс graphviz. Результат роботи програми, отриманий в консолі, можна просто надіслати на цей сервіс та отримати наочний результат та послідовність роботи всіх модулів серверної частини вебдодатку (рис. 4.3).

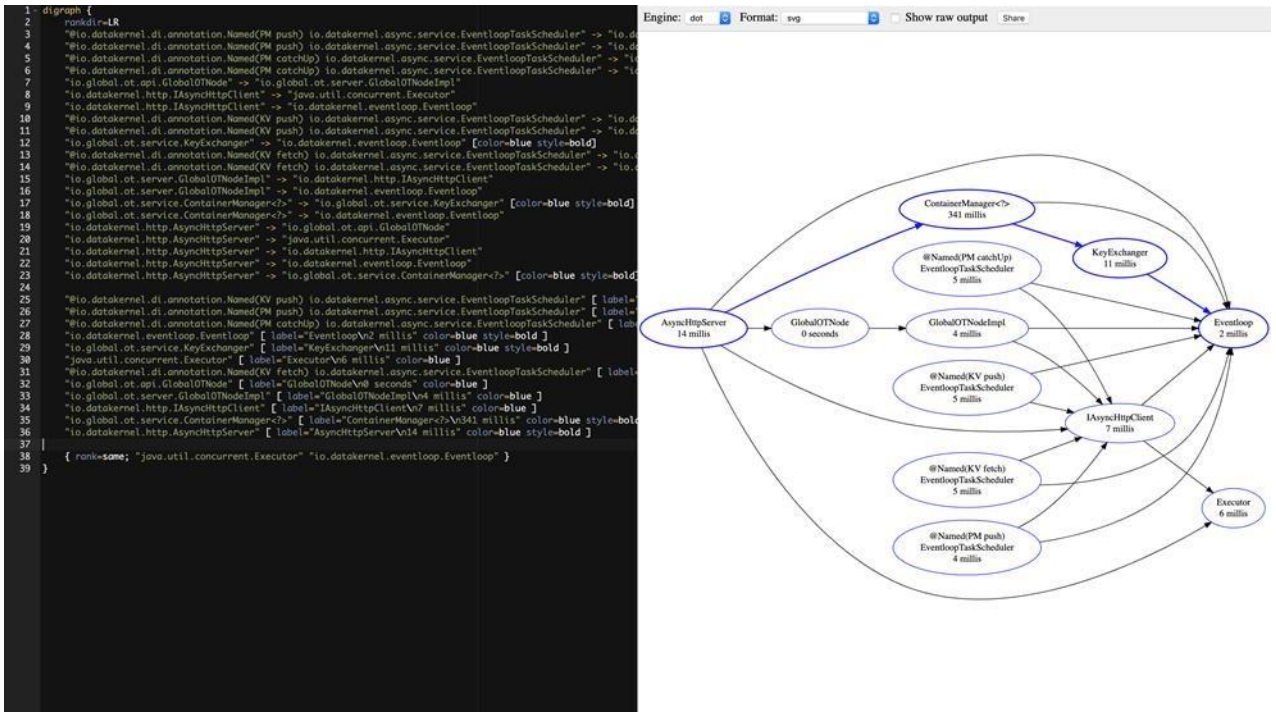

Рисунок 4.3 – Робота модулів серверної частини вебдодатку

4.2. Аналіз результатів роботи

Під час написання дипломного проєкту розроблено програмне забезпечення, яке базується на використанні операціональних трансформацій. Текстовий редактор реалізовано у вигляді вебдодатку, в якому операціональні трансформації виконуються як на клієнтській, так і на серверній частині вебдодатку.

Під час розробки даного проєкту проведено всі необхідні тести та перевірки на правильність та коректність роботи редактору текстових документів. Проміжний результат виконання операціональних трансформацій, який зберігається у вигляді графу, а також коректність та послідовність роботи модулів програмного забезпечення можна переглянути за допомогою спеціального сервісу graphviz.

ВИСНОВКИ

В ході виконання даного дипломного проєкту детально розглянуто особливості використання розподілених обчислювальних систем, операціональні трансформації на різноманітних прикладах, а також створено редактор текстових документів на основі операціональних трансформацій у вигляді вебдодатку.

У дипломному проєкті проаналізовано всі особливості та переваги розподілених обчислювальних систем, а також обґрунтовано використання РОС під час виконання обчислювальних операцій. Також розглянуто основні задачі розподілених обчислювальних систем.

Розглянуто всі основні властивості та наочно продемонстровано переваги використання операціональних трансформацій у розробці сучасного програмного забезпечення.

Розглянуто загальну структуру розробленого програмного забезпечення – його модулі, методи та особливості реалізації вебдодатку з використанням операціональних трансформацій. Особливу увагу приділено опису серверної частини вебдодатку, адже саме там відбувається основна реалізація операціональних трансформацій розробленого текстового редактору. Також продемонстровано загальний приклад роботи додатку з відповідними настановами для правильної роботи користувача. Також проілюстровано тестування розробленого додатку та показано коректність виконання всіх операцій розробленого програмного забезпечення.

Всі основні задачі, поставлені в проекті, реалізовані: вебдодаток має можливість одночасного редагування текстових файлів відразу декількома користувачами, з використанням можливостей ОТ. У разі ймовірного відключення користувача від мережі Інтернет та подальшого його відновлення вебдодаток синхронізує всі внесені зміни із сервером. Для зручного

користування розроблено зручний та інтуїтивно зрозумілий для різних користувачів інтерфейс.

Даний вебдодаток можна покращити, наприклад, додаванням автоматичної авторизації за допомогою вже створених профілів у таких відомих сервісах як Google, Facebook, Twitter і т.д.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Литвинов О.А., Хандецький В.С. - “Розподілена обробка інформації” - Д.: ТОВ «Баланс-Клуб», 2013.
2. Eric A. Brewer. – “Towards robust distributed systems. Principles of Distributed Computing” - [Електронний ресурс] – Режим доступу: http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf.
3. Andrew S. Tanenbaum, Maarten Van Steen. - “Distributed Systems: Principles and Paradigms, 2nd Edition” - Vrije University, Amsterdam, The Netherlands, 2007.
4. Ellis, C. A. and Gibbs, S. J. - “Concurrency control in groupware systems”. ACM SIGMOD (1989).
5. C. Sun, “Operational Transformation in Real-Time Group Editors: Issues, Algorithms, and Achievements” - [Електронний ресурс] – Режим доступу: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.53.933&rep=rep1&type=pdf
6. Understanding and Applying Operational Transformation - [Електронний ресурс] – Режим доступу: http://www.codecommit.com/blog/java/understanding-and-applying-operational-transformation.
7. Kumawat Santosh, Ajay Khunteta – “A Survey on Operational Transformation Algorithms: Challenges, Issues and Achievements” - [Електронний ресурс] – Режим доступу: https://www.researchgate.net/publication/45183356_A_Survey_on_Operational_Transformation_Algorithms_Challenges_Issues_and_Achievements.
8. C. Sun. – “Operational Transformation Frequently Asked Questions and Answers” - [Електронний ресурс] – Режим доступу: https://www3.ntu.edu.sg/home/czsun/projects/otfaq/.

9. Gu, N., Yang, J. and Zhang, Q. - “Consistency maintenance based on the mark and retrace technique in groupware systems” - ACM GROUP (2005).
10. Imine, A., Molli, P., Oster, G., Rusinowitch, M. - “Proving correctness of transformation functions in realtime groupware” - ECSCW (2003).
11. C. Sun and C. Ellis. - “Operational transformation in realtime group editors: issues, algorithms, and achievements.” - In Proceedings of the ACM Conference on Computer Supported Cooperative Work (Dec. 1998).
12. D. Li and R. Li. – “An admissibility-based operational transformation framework for collaborative editing systems”. Computer Supported Cooperative Work: The Journal of Collaborative Computing, (Aug. 2009).
13. Anderson, R. – “A Guide to Building Dependable Distributed Systems”. Security Engineering. - New York (2001).
14. C. A. Ellis, S. J. Gibbs, G.L. Rein – “Design and use of a group editor. In Engineering for Human-Computer Interaction. - G. Cockton, Ed., North-Holland, Amsterdam, (1990).
15. Palmer, C.R. and Cormack, G.V. – “Operation transforms for a distributed shared spreadsheet”.- ACM CSCW (1998)
16. Ressel, M., Ruhland, N. and Gunzenhauser, R. – “An integrating, transformation-oriented approach to concurrency control and undo in group editors”. - ACM CSCW (1996).
17. IntelliJ IDEA. Capable and Ergonomic IDE - [Електронний ресурс] – Режим доступу: https://www.jetbrains.com/idea/.
18. WebStorm. The smartest JavaScript IDE - [Електронний ресурс] – Режим доступу: https://www.jetbrains.com/webstorm/.